Constraint-Directed Improvisation for Everyday Activities

By

John Eric Anderson

A Thesis
Submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

ISBN 0-315-99082-1

Name JOHN ERIC ANDERSON

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer SCIENCE — SUBJECT TERM

0984 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

CONSTRAINT-DIRECTED IMPROVISATION FOR EVERYDAY ACTIVITIES

BY

JOHN ERIC ANDERSON

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Existing approaches to planning in Artificial Intelligence (such as Universal and Classical Planning) are designed for very specific types of activities, and are largely inapplicable to areas outside their narrow ranges. In particular, everyday activities that are simple for humans, such as making a meal or getting from place to place, require long-term goal-directed and timely responses that are far beyond the bounds of these traditional approaches. This dissertation examines the nature of the everyday activities and develops a computational architecture for an agent able to participate in such activities.

An analysis of everyday activities shows them to be difficult tasks made artificially simple through extensive activity-specific knowledge possessed by the agent performing them. I argue that existing approaches are unsuitable to everyday activity because they rely too heavily on compiled knowledge and fail to adequately apply the background knowledge from which these compilations were originally made. To address everyday activities, I present a theory of *improvisation*, a new approach that views the problem as satisficing intelligent control: providing resource-bounded responses to the environment in light of the agent's previous experience and its current and future intentions for activity. This process is based on the use of both heavily compiled routines the agent is accustomed to following, and an extensive collection of background knowledge used to apply those routines flexibly. The agent can rely on its routines in normative situations or when time is too scarce to spend examining the reasons behind its routines, and can conversely rely more heavily on background knowledge as situations become less normative. This allows the agent to take advantage of regularities in its environment and respond flexibly in less familiar situations.

I then present an architecture embodying the improvisational approach based on the use of constraint-directed reasoning. This methodology provides a flexible control mechanism that allows the agent to respond as dynamically as necessary for the circumstances in which it finds itself. Implemented examples of improvised behaviour are also shown, using a simulation tool developed in conjunction with this research.

ACKNOWLEDGEMENTS

There's no straight lines make up my life,
and all my roads have bends;
there's no clear cut beginnings,
and so far, no dead ends.

– Harry Chapin, *All My Life's a Circle*

It is a rare opportunity in life to get the chance to reflect on the many influences that have led you to where you are. In doing so, I am overwhelmed; while this has given me a great appreciation of the wonder and unpredictability of life in general, a list all the many acts of kindness on the part of others in my life would be much longer than this thesis and would take longer to complete. This can be only a partial list, and I am indebted to many friends and strangers that are not named here.

The time I have spent working on this thesis has sent me to many interesting places and has given me many interesting times. It has also on occasion made me tired and moody. I'd like to thank everyone around me for putting up with me during those times and excusing me when I wasn't as patient or understanding as I could have been.

I cannot think of anyone more directly responsible for my being where I am today than my advisor and friend, Mark Evans. Mark, I cannot begin to express my gratitude for all that you've done for me. Over the years you have given me opportunities that have been unimaginable to me, and have taken me to heights I never thought I could reach. Were it not for your efforts, I could never have even begun to create a work of this magnitude. You have always given tirelessly of yourself when I know you've had many more pressing things to worry about, and you always knew when to push me for more, when I needed to be left alone, and when I needed a friend. I cannot think of anyone after which I would more desire to pattern my life. You will always have my friendship and my respect. Thanks also for helping me over so many hurdles that seemed like tall brick walls over the course of this work, and for your patience in listening to interminable examples of making tea.

I also owe a great debt to my Master's advisor and friend, David Scuse. Long before I even thought myself capable of a project such as this one, you taught me how to perform research, and throughout these long years you have served as a role model for me in so many ways. To me you represent everything an academic should stand for. I admire your integrity, your

breadth, your respect for individual and community, and your love of knowledge. You are an amazing man and I will always work toward your example. Thank you for your continual efforts on my behalf and for the daily distractions that have kept me sane.

Many members of the Department of Computer Science have contributed greatly to my work and my character during the years I have been working on this thesis. Hugh Williams and Ken Barker among many others gave me much advice on academia and scientific research. Tom Dubinski and Lynne Romuld have helped me deal with millions of technical and administrative difficulties over the past five years, and deserve much thanks. I am also indebted to my fellow graduate students, and of these, Linda Strachan in particular. Linda, over the years you were always going through most of the same things I was, and you have helped me more times than I can count. Al Stephens, Richard Lukes, Renate Scheidler, Peter Graham, and Ken Hotz also gave me a great deal of advice and support.

Above all these people, however, I owe an incredible debt to my family. Without them, my life would be empty. Mom, I have learned so much from you. You taught me to rely on myself, to love learning, and you always let me know I could do anything I set my mind to. Dad, more than anyone else you have given me my character. When I hear your voice in mine, see your decisions in mine, I am always very proud. I cannot say enough how proud I have always been that you are my parents. More than anyone else have persevered to give me the skills I needed to do this and so much more. Umma, I am so happy you are here to see this, and I want to let you know how many times you have set an example for me in how live should be lived. You are a very special person. Todd and Bonnie, I love and respect you more than you could ever know. You have both given me so much over the years and I can never repay you. Suzanne, you have only been around a very little while, but you have brightened my life incredibly. I also have to thank Amaguk for waiting up many nights for me. I love you all so very much.

I also want to thank my friends for putting up with so much over all these years. I have taken you all for granted far too often. Kevin, it seems like you've always been around, and I have relied on you for so much more than digging me out of the house when I've been working too hard. I owe you a great deal. Mike and Donna, you may both be a long way away sometimes, but distance doesn't matter to the heart and it still seems like you're right next door. Steve and Michel, you have both helped me to escape when I needed it, and we've had some great times. Thank you both especially for putting up with my moodiness and giving me a bigger perspective on things. John and

Tracy have always been a great source of advice about almost anything, and have been a great help many times. Deb and Kirsten have given me many adventures and have been an outlet for my frustration more often than they deserved. Cairn Moore, Ian Ross, and the Prairie Theatre Exchange Adult Co. taught me a lot about improvisation and life, both everyday and otherwise. I would also like to thank Fiona Ruthven, Vince Wright (of waffling fame), Michelle Kim, Meg Gray, Iain Mainprize, Diane Hagglund, Trevor Moffitt, Paula Piilonen, Carrie Miller, Tom Wallace, Sanjay Patil, Shahid Nimjee, Cheryl Pinkney and Tyler Rosewood. Your thoughts, letters, phone calls, and Email have sustained me through many dark days. You've all been wonderful friends, and I thank you for being there when I needed you.

I also wish to thank all of my students over the years I have been teaching. Knowing that I have in some small way made a difference in your lives has meant the world to me. Teaching during the entire time I have worked on this thesis has been the major reason that this work has taken the amount of time it has. However, I wouldn't have given up a moment of it. I'd like to sincerely thank Ralph Stanton and Peter King for giving me the opportunity to teach over the course of this work, and to all the members of the Department of Computer Science for treating me as a colleague as well as a graduate student.

Parts of this work and the ideas behind it were discussed with many people, and I have benefited from the feedback I have been given by many reviewers and conference participants. Thanks also to all who sent me source code and documentation for many of the works described herein. Jacky Baltes was a big help during early work that led to this thesis and was also a great travelling companion to many conferences. Jerzy Rozenblit and Witold Pedrycz also served on my committee and I thank them for their efforts at making this a better thesis.

The National Science and Engineering Research Council supported for the first year of my doctoral research. The University of Manitoba provided generous support for most of the remainder in the form of a University of Manitoba Fellowship.

Finally, I wish to thank an institution and the people involved in it for making me the person I am today. That institution is Shad Valley, and I owe it and the people involved in it with me more than I can ever describe. Shad Valley came along at a crucial point in my life, and made me aware of who I am and the capabilities I had in me. You all gave me the opportunity to speak with a voice I never knew I had before, and also the opportunity to begin to

believe in what I was saying. I have discovered so much of myself I never knew existed because of all of you, and you have changed the direction of my life beyond my wildest dreams. For this I am in humble gratitude. Among this group are many of the people I have already named, but I wish to add Dean Kriellaars to this list. Thanks for many enlightening discussions and much advice about life. Thanks also to everyone whose name is in the blue book that reassures me when I forget how fortunate I really am.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

> Psychologists have chronicled the failures of thought, the nonrationality of real behaviour. Even simple tasks can sometimes throw otherwise clever people into disarray. Even though principles of rationality seem as often violated as followed, we still cling to the notion that human thought should be rational, logical, and orderly. Much of law is based upon the concept of rational thought and behaviour. Much of economic theory is based upon the model of the rational human who attempts to optimize personal benefit, utility, or comfort. Many scientists who study artificial intelligence use the mathematics of formal logic - the predicate calculus - as their major tool to simulate thought.
>
> But human thought - and its close relatives, problem solving and planning - seem more rooted in past experience than in logical deduction. Mental life is not neat and orderly. It does not proceed smoothly and gracefully in neat, logical form. Instead, it hops, skips, and jumps its way from idea to idea, tying together things that have no business being being put together; forming new creative leaps, new insights and concepts. Human thought is not like logic; it is fundamentally different in kind and in spirit. The difference is neither worse nor better. But it is the difference that leads to creative discovery and great robustness of behaviour.
>
> – Donald Norman, *The Psychology of Everyday Things*

1.0. Introduction

This dissertation is about the processes and mechanisms that allow an intelligent being to go about everyday activities - commonplace activities such as making breakfast, driving to work, or doing the dishes. Humans perform these everyday activities routinely, and with great ease. In spite of this ease however, those interested in creating intelligent computer systems have been

confounded by these phenomena to date. While Artificial Intelligence (AI) has been successful at creating intelligent systems that emulate very detailed and formal types of problem-solving activities (e.g. medical diagnosis [Shortliffe et al., 1979], molecular genetics [Stefik, 1981a, 1981b]), everyday activities remain an enigma. Even outside of AI, everyday activities themselves are poorly understood, especially when compared to the extensive work available on more specific and intuitively complex activities (e.g. mathematical problem-solving [Polya, 1957], medical problem-solving [Cutler, 1985]). This dissertation addresses everyday activities through an examination of their characteristics, an examination of the inadequacies of previous approaches, and the design and implementation of an intelligent system specifically intended for engaging in everyday activities.

1.1. Motivation: Artificial Intelligence and Planning

> Shakespeare created nothing. He correctly observed, and he marvelously painted. He exactly portrayed people whom *God* had created; but he created none himself. Let us spare him the slander of charging him with trying. Shakespeare could not create. *He was a machine, and machines do not create.*
>
> – Mark Twain, *What is Man?*

A concise and generally accepted definition for the field of Artificial Intelligence (AI) has always been an enigma: in fact, AI is a field with many definitions and many goals. Consider some of the ultimate purposes that have been attributed to artificial intelligence in the past:

- to endow machines with reasoning and perceptual abilities [Webber and Nilsson, 1981];
- to build an intelligent machine and find out more about the nature of intelligence [Schank, 1987];
- to provide the foundation for a system capable of general intelligent behaviour [Laird et al., 1987];
- seeking to understand and build fully human-like systems [Covigaru and Lindsay, 1992];
- to make machines more useful [Winston, 1984];
- to create computer programs that are capable of acting intelligently in the world [McCarthy and Hayes, 1969].

While these goals are wide ranging, there is a collective theme among them: the construction of autonomous agents[1] that display flexible, rational, intelligent behaviour. The ultimate goals of AI concern understanding the concepts of intelligence and embodying these concepts in computer systems to allow them to deal with problems that are beyond the scope of conventional computer systems.

The difficulty in providing a definition for the field of artificial intelligence can be blamed in part on the wide scope of the field. Behaving flexibly involves many individual contributions, such as being able to perceive, being able to reason logically, being able to relate past experiences to new situations, and being able to learn, just to name a few. These are all components of intelligence, but no one of these features would suffice to allow an entity to be considered intelligent in the usual sense of the word. Because of this wide range, the criteria that are used to define intelligence are also widely debated. A number of summaries of such criteria exist (e.g. [Schank, 1987; Newell, 1989; Ackerman, 1992; Sternberg and Salter, 1982]), and the criteria themselves range from the ability to apply old experiences to new situations, to the ability to communicate ideas. Sternberg and Salter [1982] argue that despite the wide variation in these criteria, they share a common thread: the ability to perform both adaptive and goal-directed behaviour. They describe this relationship as follows [pp. 24-25]:

> If there is a common theme in the highly diverse viewpoints presented...it is that intelligence is expressed in terms of adaptive, goal-directed behaviour...Adaptive behaviour is behaviour that confronts and meets successfully the challenges that are encountered....Goal-directed behaviour is behaviour that is ultimately purposive. It is not enough for behaviour merely to be adaptive: it is possible to adapt to the demands in an environment so as to do nothing more than minimally "get by". Intelligent behaviour must be understood in terms of the short- and long-term goals to which it contributes. The meeting of challenges cannot be termed "intelligent" if the performance involved in meeting these challenges is aimless.

The research presented in this dissertation falls into the area of artificial intelligence that has come to be known as *planning*. At its most basic, planning involves finding a sequence of actions (a *plan*) to satisfy some goal or desire, and then applying that plan in some environment to bring about the desired goal[2]. Even a definition as general as this would leave many

---

1 Embodied minds that can affect the world around them according to their own decisions.

2 For example, deciding on a route to travel to go to the store, and then following that route, step by step.

planning researchers unsatisfied. Like AI in general, a study of any recent cross-section of planning research (e.g. [Hendler, 1992]) reveals a diverse collection of views extending even as far as a basic definition of the field.

Planning as a field involves issues at the core of the general goals of AI presented above. Planning involves the creation of agents that can go about the world and alter it through their reasoning and behaviour. Core research in planning involves answering questions such as:

- How does the environment in which an agent is expected to perform affect the design of such an agent?
- How does an agent decide what is a rational course of action to adopt in a specific situation?
- How should knowledge about the actions an agent can perform and the experiences it has been through be organized and accessed?
- How should information from external sources be integrated with the agent's lines of reasoning?

Agents that can rationally alter the world require many capabilities: the ability to reason about action and make decisions about courses of action; to perceive the world around themselves and to understand those perceptions; to reason about others in the world and the effects they will have on the agent's intended courses of action; and to reason about how the world works. However, even though recent research has shown more and more overlap between these areas, planning is normally concerned with the central issues of action and decision-making, rather than peripheral issues such as perception.

Planning has always been a major focus of research in artificial intelligence. The emphasis placed on this area is due in part to the historical goals of artificial intelligence. However, research in this area has also gained importance because of an overlap with many other relevant fields within AI. The reasoning mechanisms required by a successful planner include temporal and spatial reasoning, for example. Indeed, while few planners are designed completely around reasoning about time (a well-known exception being [Vere, 1983]), planning has often been equated with temporal reasoning in that it involves an agent intelligently reasoning about the future and altering that future using its knowledge of the actions it can perform [Tsang, 1988; Anderson, 1991].

Many other fields of AI also have relevance to planning: real-time processing and perception-related fields such as computer vision and speech understanding, for example. In fact, many fields that bear no obvious connection to the central issues of planning can be classed as planning problems. Problems in natural language understanding can be viewed as planning problems where the planned actions are communications [Cohen and Perrault, 1979]. Plans as sequences of actions are also used in other forms of understanding problems: story and scene understanding, for example [Wilensky, 1983]. Current emphasis on the ability of a system to perform autonomously as a primary characteristic of intelligence [Barker et al., 1992; Covigaru and Lindsay, 1991] is also resulting in further emphasis on planning in the AI community.

While research in planning is voluminous, it is mainly concerned with one of two foci: *classical planning*[3], where completely detailed plans (similar to computer programs) are constructed and then executed (again, like a computer programs) at a later time; and *universal planning*, a more recent theory wherein an agent continuously reacts to individual situations rather than making use of long-term plans. In particular, the classical planning view is so pervasive that the term planning itself has been traditionally associated with the classical planning perspective. Indeed, the predominant view of all human activity for many years was one of complete control through the creation and execution of plans [Miller et al., 1960; Newell and Simon, 1972].

In recent years, however, the idea of planning has come to mean much more than the creation of a plan followed by its execution in a computer program-like manner. This has resulted from the increasing realization that much of activity is beyond the strict application of plans. Some activities employ no planning whatsoever (e.g. it is hard to envision picking up an extremely hot cup, and then dropping it because one realized that it was burning and constructed a plan to open the fingers of one's hand). Other activities make use of plans, but do not follow them in detail (e.g. one might plan one's pace in a marathon, but not where one intends to place each step). Historically, however, research in these areas is still known as planning, despite the fact

---

3 In a field still in its infancy, the term *classic* may seem a misnomer. The term "classical planning" comes not from a generally accepted or "classic" method of performing planning, but from the techniques of classical logic on which the approach is based. These planning models are described and analyzed in detail in Chapter 3.

that detailed plans may not play a strong role. A more modern definition of planning is making use of any type of construct that constrains or provides guidance for the actions of an intelligent agent [McDermott, 1991].

The two planning perspectives described above allow agents to perform very well in certain specific situations. Classical planning is useful in situations where actions can be computed in advance and the environment is otherwise static (e.g. it could be used to produce an advanced route plan for a road trip, if that route could be followed exactly). Universal planning works well in areas where every situation the agent could arrive upon has a single logical response, allowing the agent to rely solely on its senses to take the next logical step at any moment (e.g. many video games). However, agents operating from these perspectives perform poorly in domains that fall outside these strict realms.

This dissertation argues that everyday activities lie outside both these realms, and that using these planning models as a basis for dealing with everyday activities is misguided: a more appropriate agenda is the examination of everyday routines and the construction of a model particularly designed to suit these phenomena[4]. The thesis of this dissertation is that humans improvise in their performance of everyday activities: that is, they use detailed routines in commonly-practiced activities, and make use of the background knowledge behind those routines to apply them flexibly in varying situations. I develop a model of improvisation for an intelligent agent, and illustrate the model in varying instances of a particular everyday activity.

1.2 . On the Nature of AI Research

When performing or evaluating research in artificial intelligence, one must remember that it is a very young science. While people have dreamed of intelligent machines in one form or another since the dawn of time, it has only been in the last half of the twentieth century that those dreams have begun to be realized, or even taken seriously.

---

4 The reader will see in Chapter 3 that this technique is precisely the reason that Classical and Universal planners are so well-suited to the domains they occupy.

Research in an immature science such as artificial intelligence[5] is inherently different from that of a mature science such as physics or chemistry. In the latter, new discoveries are usually comparatively small modifications to a widely accepted (though not necessarily proven) model or theory that explains much of the field. On rare occasions, large parts of the theory are called into question (e.g. quantum vs. Newtonian mechanics, or Ptolemaic vs. Copernican views of the universe), but in general, these fields enjoy the benefit of a large body of accepted truth on which to base future work. In an immature science such as AI, this is not the case. While a few significant works have defined a perspective for viewing portions of the field (e.g. [Minsky, 1986; Newell and Simon, 1972]), there is not yet any generally accepted theory or framework on which to base ongoing research. The result of this is that researchers in learning, for example, cannot make use of a generally accepted theory of planning to qualify or simplify their research. Similarly, researchers in vision cannot make use of a generally accepted theory of learning. In both of these cases, any overlap from one subfield to another entails making assumptions of unsolved problems in the other subfield (qualifying and possibly compromising one's own research) or embarking on a number of potentially extremely time-consuming research projects in order to provide a more solid foundation for the original research. The former is most often chosen over the latter for reasons of expediency and resource limitations, and because of this the field of AI as a whole often appears rampantly unscientific. In those cases where assumptions are made without benefit of explanation or analysis, the research involved *is* rampantly unscientific. However, these are isolated cases rather than a general trend, and I believe that such cases will naturally diminish as the field becomes more mature and has a stronger theoretical basis.

This dilemma arises in virtually all AI research, and few areas suffer more than research involving intelligent agents. This area is affected particularly severely because the creation of complete, autonomous intelligent agents is at the core of AI. As such, almost any research project in this area involves a great deal of overlap with other areas (e.g. vision, knowledge representation, robotics, etc.), with the necessity of solving open problems in those areas or making assumptions about the nature of those problems. Thus, if one develops a theoretical architecture for an intelligent agent, unless one intends to solve every open problem in areas such as computer vision and robotics,

---

5 Putting aside debate on AI as a field of science, engineering, or philosophy.

one is forced to make (possibly incorrect) assumptions about how these various peripheral aspects will operate when more complete theories are available. The focus of an individual research project in intelligent agency may be quite narrow (e.g. demonstrating the utility of a new form of representation within an intelligent agent). In such cases, it is almost completely irrelevant to provide complete answers to peripheral problems such as sensing. These peripheral aspects are by no means independent, but can be de-emphasized as long as the assumptions made to deal with them are explicit and reasonable, and that results from such research are interpreted in light of those assumptions.

The number of peripheral issues that must be dealt with in any immature science is large; however, the reader should not assume from this that work in an immature science has only negative aspects. While special considerations must often be made, research in an immature science is at the same time refreshing and invigorating, in a way that no mature science can duplicate. Ginsberg [1993] describes this excitement eloquently:

> Imagine what it must have been like to be a physicist in Newton's day...From a modern point of view, they didn't know *anything*. The problems and challenges faced by a physicist in the sixteenth century weren't any easier than those faced today, but they were tremendously more *accessible* - it's a lot easier to explain $F = ma$ to a person on the street than it is to explain the intricacies of quantum field theory. It was an age of scientific romanticism...Artificial intelligence is like this today. It is a science in its infancy, and that makes it special. Perhaps the most important way in which AI's newness as an enterprise makes it different is the fact that it shares an excitement with the physics of four hundred years ago. Newton described himself as a child playing on a beach, making progress simply by picking up interesting-looking stones and examining them; one has the feeling, heading his description, that physics was so full of exciting problems that he found it difficult to know where to start.

Like any science, AI involves making models to explain specific phenomena: the phenomena of intelligence. Also like any science, models are never completely correct or incorrect. A model explains a set of phenomena to a given degree, and it supplants other models that do not provide as good an explanation. In this sense, a scientific model is never wrong: it is merely improved upon. Thus Newton was not proved wrong by Einstein; rather, Einstein's model of physics provides a better explanation of a specific set of phenomena than did Newton's. In the same sense, nothing in this dissertation should be taken as intending to prove wrong any earlier work. In Chapter 3, for example, I will argue that classical and universal planning do not explain the phenomena of everyday activities well. This does not mean that these models are wrong, or that they do not adequately explain other

phenomena. Rather, it means that there are aspects of activity that require a different model to explain.

AI is also different from many other sciences in that because it is largely a computationally-based field, its models may be implemented. This adds a new facet to the scientific models of AI and those derived in fields such as physics or psychology. While all of these models are used to explain specific phenomena, those used in AI are more than simply an abstract set of rules or guidelines. They can and should exist as physical implementations: artifacts that not only passively explain a set of phenomena, but actively demonstrate them.

1.3. Overview of This Dissertation

This dissertation is divided into nine chapters. All are, of course, highly interrelated. However many describe aspects of this research that also stand well on their own.

Chapter one introduces AI and planning, and explains the motivations for this research.

Chapter two analyzes the nature of everyday activities, differentiates them from other types of activities, and provides a list of characteristics that an agent designed to participated in everyday activities must embody.

Chapter three presents an overview of existing planning perspectives, and analyzes them from the perspectives of the characteristics derived in Chapter two.

Chapter four presents a theory of improvisation, an approach used by humans to participate in everyday activities, and describes the utility and requirements of improvised behaviour.

Chapter five presents an architecture for an agent capable of improvisation, including methods of resource-bounded control of reasoning.

Chapter six describes a simulation system designed not only to support an implementation of such an agent, but to support general experimentation with intelligent agent designs in varying environments.

Chapter seven describes a partial implementation of this architecture and illustrates various examples using the simulation system presented in Chapter six.

Chapter eight relates this work to previous and current projects elsewhere.

Chapter nine presents an analysis of the research presented in this dissertation, describing its major contributions and limitations, and also outlines future work.

CHAPTER 2

EVERYDAY ACTIVITIES

Everyday (adj.) Pertaining to daily or common life or occasions; used or occurring habitually; suitable for or that may be seen every day; common; usual.

– *The Century Dictionary*, 1895

To know
That which before us lies in daily life,
Is the prime wisdom.

– John Milton, *Paradise Lost*

The whole of science is nothing more than a refinement of everyday thinking. It is for this reason that the critical thinking of the physicist cannot possibly be restricted to the examination of the concepts of his own specific field. He cannot proceed without considering critically a much more difficult problem, the problem of analyzing the nature of everyday thinking.

– Albert Einstein, *Ideas and Opinions*

2.0. Introduction

Humans participate in a wide range of activities in an average day. Many of these are simple, mundane activities requiring little thought, such as walking the dog or preparing a simple meal. Others require varying amounts of concentration, such as balancing a chequebook, preparing a budget, or planning a class schedule.

AI researchers have at one time or another been interested in all of these types of activities, and many others as well. Early AI research concentrated

largely on activities that fall into the latter group: those that normally require great concentration in humans, and are at the limits of human mental capabilities [Wilensky, 1983]. For example, the General Problem Solver (GPS) [Ernst and Newell, 1969], a direct ancestor of most modern planning systems, was based on a series of studies of human reasoning in solving logic, chess, and cryptarythmetic problems [Newell and Simon, 1972]. The types of problems that these systems dealt with have come to be known as *micro-worlds* [Drefus, 1981], since they abstract many of the difficulties encountered in everyday reasoning and allow the system to focus on other aspects of problem-solving.

Later systems, such as the ISIS system for factory scheduling [Fox, 1983] performed successfully in much more dynamic and less well-understood domains. However, these systems still concentrated on the extreme types of problem-solving activities studied by GPS and related projects: reasoning that is significant in many forms of advanced problem-solving, but which has little bearing on many of the tasks humans perform in an average day. Humans undoubtedly perform the types of tasks demonstrated by systems such as ISIS, and employ some of the same reasoning mechanisms. However, such activities require extensive mental (complete concentration over significant periods of time) and physical (pen and paper, calculator, etc.) resources to complete. Such activities can be categorized as *thinking in closed-world systems*, an uncommon form of human cognition [Barker, 1968].

The types of activities we commonly call *everyday* activities are of a fundamentally different character than this. Everyday life is not a cryptarithmetic problem. An activity such as washing dishes seems, on the whole, to have little to do with the reasoning mechanisms used to play chess. Travelling from home to work has few commonalities with logic problems. The types of activities in which the average person participates often, such as making a meal or going to work, are performed easily and naturally. Little in the way of deep concentration is required; certainly nothing on the order we would associate with solving logic problems or constructing optimal schedules for machine-shops.

Despite the ease with which humans go about everyday activities, they are still objectively very complex - as much so, I argue, as the logic and optimization problems discussed above. The everyday tasks we perform easily take place in a world that can be difficult to understand and difficult to predict. Our everyday activities are also diverse and continuous: there is almost always *something* to be done. Thoreau lamented about the ever-increasing complexity of mankind's everyday activities in 1847 [Atkinson,

1992], and few would disagree that the diversity of everyday tasks and the knowledge necessary to perform those tasks has increased greatly since then.

In addition to being complex, the world around us is often dangerous as well. When one examines any commonplace activity, one can usually think of numerous misadventures that could bring harm to a participant. For example, consider making tea: one could leave the stove on, have loose clothing set afire by the stove, spill hot water on oneself, or any number of things. Even crossing the street presents its dangers, as Hammond's [1989a] well-known Yale robot joke illustrates:

> It was the year 1995 and the DOD finally decided that it wanted to see some results. So the head of research called together all of the scientists doing work in robot planning to demonstrate their work. Everyone was faced with the same task – get a robot to cross a busy street.
>
> The first robot was a first -principles planner from Stanford. After walking to the curb, it looked left, then right and sat for 20 minutes planning and projecting until it found a provably correct plan of action. Once finished it stepped into the road, screamed, "I thought this was a closed world", and was hit by a speeding truck.
>
> The next robot was the JPL Mars Rover. Integrating planning and execution, it began to slowly step out into the street, probing the ground every two feet for "deadly Martian dust pits". For ten hours it probed and stepped, stepped and probed. Eventually, the sun set, the solar powered Rover ground to a halt and was immediately hit by a speeding truck.
>
> This was quickly followed by the CMU robot pogo stick that jumped madly about, dodging trucks left and right until it hopped happily into an open manhole.
>
> After this came the MIT situated activity device. It leapt into the road and deftly skipped between cars. It dodged traffic impressively until, pinned on each side by ice trucks, it was nailed by a van from Bee Line Movers.
>
> And so it went. The day wore on, and after countless tries, no robot made it across the street. Finally there was only one robot left – the Yale robot.
>
> Unlike the others, it was not clean and shiny. It was not silvery and smooth. It was dented and bashed, with little to be said for it. But still it bravely moved up to the curb.
>
> It looked at the battered bits of steel and wire in front of it. It listened to the faint echoes of the now distant robot pogo. It looked at the expectant faces in the crowd.
>
> Then it paused, put its scruffy robot hands in its scruffy robot pockets, eased back on its rear wheels and said, "You know, this reminds me of a story..."

This joke is obviously intended for a very specific audience, and there is much subtle humour in it that will only be evident to those well-versed in

the AI planning literature[6]. The Yale robot joke does much more than poke fun at perceived inadequacies of current trends in AI research, however. It demonstrates that many of the activities that we take for granted as simple and relatively harmless are in fact fraught with danger to one inexperienced with them. Somehow, humans find their way through this danger and complexity, and individual everyday activities are accomplished (usually) easily and successfully. In fact, humans function not only adequately but *well* in the midst of this complexity and danger.

Interest in everyday activities has increased greatly in recent years because of two reasons. As mentioned above, it is increasingly being accepted that the early forms of reasoning in problem-solving activities do not accurately reflect the diversity of human cognitive mechanisms[7]. From a cognitive science perspective, a better understanding of human cognition requires a better understanding of the reasoning involved in performing day-to-day tasks.

From an AI perspective, the relative ease with which humans perform complex everyday activities and the relative lack of success of AI systems to duplicate such performance has always been an dilemma. Despite the fact that reasoning about the world in a commonsense way has been one of the goals of AI since its inception [Dehn and Shank, 1982], the great successes of AI to date have always laid in other areas. Formalizing and computationally reproducing expert reasoning, for example, has been one of the great success stories of AI. Yet for all their success, expert systems remain brittle, in that they perform poorly outside of a extremely narrow range of tasks, and unscalable, in that the techniques they employ have been largely unadaptable to larger or more general tasks [Buchanan and Smith, 1989]. Studying everyday activities and the types of reasoning behind them may shed new light on the factors behind this contradiction, and may ultimately offer a solution to it.

---

6 Many of the AI planning perspectives involved and their relationship to everyday activity will be discussed in Chapter 3.

7 In fact, current research is demonstrating more and more that even the strict, rigourous techniques characteristic of AI problem-solving do not apply to human problem-solving in most situations; and that problem solving is "a lively, seemingly disordered though highly intellectual process, rather than a rigid structured one" [Cutler, 1985].

2.1. Defining Everyday Activity

One of the foremost problems for those attempting to examine the nature of everyday activity is providing a definition of just what makes up an "everyday" activity, and what differentiates it from other types of activities. While many in AI have claimed to operate in the realm of "everyday life" [Agre, 1988] or the "commonsense domain" [Alterman, 1988], no one has yet produced a satisfactory definition of what constitutes such behaviour. Most definitions are extremely colloquial, such as that of Agre [1988]:

> Everyday life is just the whole of our ordinary activity: making breakfast, gardening, hanging out, walking into town, cleaning up messes, shopping in supermarkets, working in offices, participating in everyday rituals.

There are many reasons for such colloquialism. Attempting a mathematical definition of a qualitative concept such as "everyday activity" is like providing a mathematical definition of love or laughter. Formally analyzing everyday activity in the manner that one would analyze the computational complexity of an algorithm is impossible because of the very nature of everyday activities. However, there is much more to be said about everyday activities than colloquial definitions such as the one above encompass.

While philosophers have examined the nature of activity in general, the most relevant modern work on characterizing everyday activity has been performed by design theorists, specifically those who design tools and objects to enhance or simplify everyday activities. In particular, Norman [1988] agues that the apparent simplicity of everyday activities lies in the nature of the tasks involved. As opposed to complex tasks such as chess, where the overall decision structure of the problem is wide and deep (i.e. a decision tree showing possible moves and counter-moves is wide and deep), Norman defines an everyday task as one with a narrow or shallow decision tree. Two examples of such structures are illustrated in Figure 2-1. As an example of an activity with a narrow structure, Norman uses the example of starting a car. Even though there is a significant number of actions that must be performed in a particular order, there are few alternatives to consider at any point. This is due mainly to the design of the automobile: the starter switch cannot be turned until the key is inserted, the transmission cannot be engaged until the car is running, and so on. Shallow structures, on the other hand, are represented by tasks such as selecting a flavour of ice cream at an ice cream parlour: there may be many choices, but only one level of decision-making [Norman, 1988]. Similarly, Norman defines non-everyday tasks as those with wide and deep decision trees, requiring considerable planning and conscious effort to accomplish.

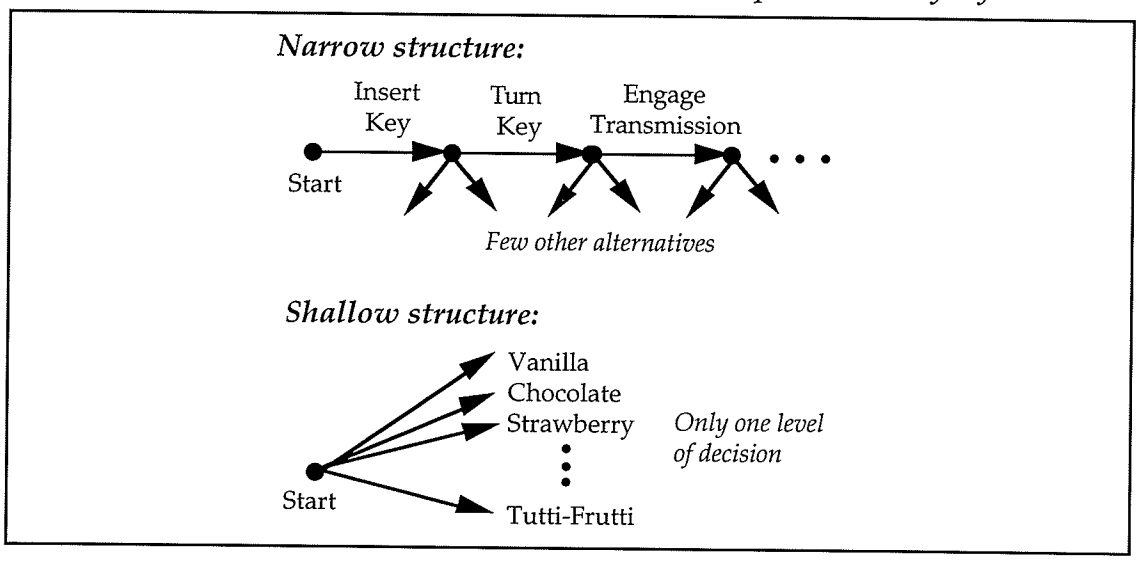

Figure 2-1. Examples of activities with narrow (above) and shallow (below) structures. Original examples, [Norman, 1988].

Norman's definition is useful from the point of view of design, but it is inadequate from an AI perspective in that it fails to consider the properties of everyday activity that give rise to shallow and narrow decision structures. Simply stating that shallow or narrow decision structures are the norm for everyday activities does not greatly aid in the study of mechanisms for recreating these behaviours. Moreover, this definition also does not distinguish between the common everyday activities mentioned above and the many micro-worlds that were common to early AI research. For example, simple logic problems (and classic AI examples) such as the missionaries-and-cannibals problem can be considered to have reasonably narrow or shallow task structures, but are by no means everyday activities.

The properties that lie behind the shallow and narrow decision structures that characterize everyday activities are encompassed by the dictionary definition quoted at the beginning of this Chapter. First, everyday activities are *common:* they occur frequently. This implies that the agent performing such activities is well-experienced with them, and possesses a great deal of knowledge about how an activity should proceed and the objects and interactions involved in it. Everyday activities are also *mundane* or routine: by virtue of an activity being of the everyday variety, there are few if any surprises to be expected while performing it.

The use of the word *common* in this definition also has a secondary connotation: that of less-than-perfect. The idea that human beings produce satisfactory responses to many problems rather than consistently optimizing

their performance is well-known. Simon [1982, p. 259] refers to this process as *satisficing*:

> It appears probable that, however adaptive the behaviour of organisms in learning and choice situations, this adaptiveness falls far short of the ideal of "maximizing" postulated in economic theory. Evidently, organisms adapt well enough to "satisfice"; they do not, in general, "optimize".

However, only certain types of activities tolerate satisficing well. In particular, activities normally performed by experts in their field (e.g. a doctor, an engineer, a lawyer, or any other trained professional) require optimal or near-optimal performance. Experts must generally guarantee results, and the consequences of not meeting this guarantee are usually significant: patients die, buildings collapse, and innocent people go to prison. Indeed, the number of lawsuits and inquiries into the conduct of professionals seen in the daily papers lends great testimony to the strength of the guarantee we expect from expert behaviour. This principle holds far beyond these extreme examples: a professional chef, for example, would not last long in a restaurant if even one in ten meals he or she prepared was not close to perfection.

In everyday activities, however, the consequences of less-than-optimal performance are usually minimal, making satisficing an appropriate approach. When I make tea, for example, the tea is still drinkable if I leave the tea bag in a little too long, or if I accidentally put slightly more milk than I usually like in it. Even when performing more obviously dangerous activities such as driving a car, less-than-optimal performance is the norm: rarely does any driver steer a car in a *completely* straight line and make each turn in absolutely perfect form, for example. There are, of course, limits to the level of sub-optimality that is acceptable: I could not simply close my eyes and turn the steering wheel at random while driving, for example.

Satisficing is not merely appropriate to everyday activities: *it is crucial*. In the real world, any agent is *rationally bounded* in that both the information an agent can absorb and the control that agent may wield over its environment is limited [Simon, 1957][8]. The optimal performance of everyday activities directly contradicts this concept: I have already described the complex nature that underlies the everyday activities that humans perform with ease, and the fact that one has a limited time to deliberate before taking action in any

---

8 The concept of rationality in everyday activities will be explored more fully in Section 2.3.

situation makes it physically impossible to always choose the optimal alternative. Simon [1969] expresses this argument well:[9]

> Normative economics has shown that exact solutions to the larger optimization problems of the real world are simply not within reach or sight. In the face of this complexity the real-world business firm turns to procedures that find good enough answers to questions whose best answers are unknowable. Thus normative microeconomics, by showing real-world optimization to be impossible, demonstrates that economic man is in fact a satisficer, a person who accepts "good enough" alternatives, *not because he prefers less to more but because he has no choice.*

2.1.1. A More Concise Definition

The common element behind the aspects of everyday activities described above is *experience.* The fact that the activity has been experienced many times means that the vast majority of possible surprises have already occurred, and the agent has the ability to deal with them should they arise again. If something happens that the agent has not experienced, then by definition the activity is no longer an everyday activity. The reason why the decision trees of everyday activities are shallow and narrow is because the agent's knowledge about these activities imposes this structure (Figure 2-2).

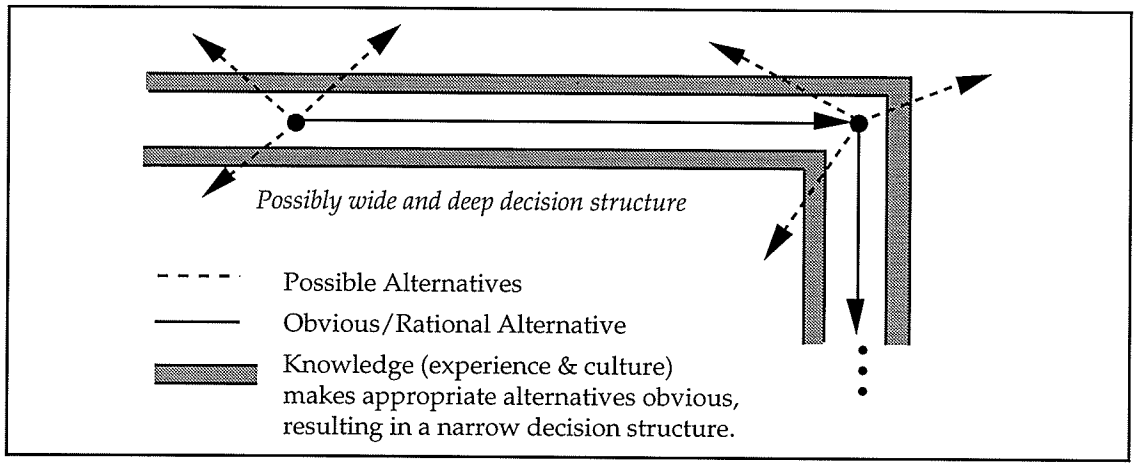

Figure 2-2. Knowledge makes useful alternatives obvious despite deep and wide decision structures

Because an agent has experienced the activity many times, the agent does not have to reason about most of the alternatives available at any one time: it

---

9 Emphasis mine.

knows the alternatives that have worked before, and illogical or irrational choices are simply not considered. *That is, the agent's knowledge acquired through performing the activity many times previously serves as a constraining mechanism on decision making.* This is illustrated by the heavy grey framework around the decision structure in Figure 2-2. This is not meant to imply that knowledge of the problem forms a physical barrier that always allows only certain alternatives to be considered. Rather, the agent's previous experience of the activity acts as a guide that reminds the agent of alternatives that are likely to be useful and limits consideration of inappropriate or irrational alternatives.

In general, the application of previous experience and activity-specific knowledge allows an agent to tackle complex everyday activities with ease. This distinguishes everyday activities from the micro-worlds or toy domains common to early AI systems. The reason that problems in the latter category (e.g. the missionaries-and-cannibals problem) have reasonably narrow or shallow task structures is that they are oversimplified problems to begin with, and bear no relationship to everyday activities. This fits Newell and Simon's [1976] description of simple and hard problems:

> What makes a problem a problem is not that a large amount of search is required for its solution, but that a large amount *would* be required if a requisite level of intelligence were not applied.

Knowledge is the key to successfully performing everyday activities in a complex and dangerous world. Some of this knowledge is acquired by direct experience: making tea in many different places on many different occasions, for example, allows us to learn about where tea-making can go wrong[10] and what parts of the world that tea-making interacts with are dangerous. Trying to make tea with *no* knowledge whatsoever about any of the actions or components that make up tea-making would be difficult if not impossible, and also extremely dangerous. In fact, with absolutely no knowledge about tea-making, the very idea of *wanting* tea would be next to impossible: that it would come naturally to people in an isolated context to brew leaves in hot water to obtain a beverage is unlikely. Much of this knowledge is not discovered through specific episodes of the activity, but supplied by culture[11]

---

10 Hammond [1989a] has extensively chronicled this type of learning.

11 That is, learned from other individuals or somehow indirectly supplied by others in our social world. Of course, all cultural knowledge must at some point have been discovered by individuals and added to the base of cultural knowledge. However, the contributions of any one individual compared to the overall collection of cultural knowledge will always be comparatively minor: individuals base their small contributions on the collective

(continued)

[Goffman, 1974; Agre and Horswill, 1992]. Indeed, culture can be defined by its psychological effects on the actions of individuals [HumCog, 1981]. In the case of dangerous situations, knowledge obtained from culture becomes much more important than direct experiential knowledge, since the inherent danger in the domain prevents extensive experimentation.

While the utility of experiential knowledge in everyday activity is important, it is also important to recognize that not all of it is learned factual knowledge, stored away and recalled as necessary. While this makes up a great deal of experiential knowledge, humans also make use of a wide array of knowledge that is easily obtainable from the world around them: "knowledge in the world" [Norman, 1988; Chapman, 1990; Agre and Horswill, 1991] as opposed to "knowledge in the head". Tools and machines, for example, have evolved for a particular use, usually having been altered and improved upon many times. Automobiles are a good example of this. The activity of operating an automobile has by no means always had the narrow structure that it currently possesses: the examples of good design cited in the previous Section were all instituted gradually, in order to make driving a more stable and everyday task. The general task of driving is another example. Commonly travelled roads are constructed in such a fashion that little mental effort is required while driving at an appropriate speed, and most required information (for moment-by-moment and large-scale navigation) is posted on easy-to-understand signs [Miller, 1994]. Indeed, it has been shown [Hammond et al., 1992] that in many ways, humans gradually make commonly occurring tasks more stable through modifying the environment in such a way that the task is performed with minimal cognitive effort: everything from placing often-used objects in convenient locations to creating tools to satisfy a particular function.

Because of the crucial nature of experiential and cultural knowledge in understanding everyday activity, capturing all three aspects of the previously quoted dictionary definition within an AI context is important. An *everyday activity* is:

a) A complex task that is both common and mundane to the agent performing it;

---

knowledge supplied by their predecessors, and often overlook this reliance. The influence of culture on activity is important, and will be described further in Chapter 4. The interested reader is referred to Mark Twain [1917] for an intriguing interpretation of these ideas.

b) One about which an agent that has a great deal of knowledge, which comes as a result of the activity being common, and is the primary contributor to its mundane nature; and

c) One at which adequate or satisficing performance rather than expert or optimal performance is required.

The common and mundane nature of everyday activities has been emphasized before in architectures for AI, especially by Agre [1988] and Chapman [1990], but not expressed as part of the definition of the concept itself. This definition includes the crucial aspects of everyday activity discussed above, and also excludes tasks that are simple enough to have naturally-occurring narrow or shallow structures. Thus it excludes the many micro-worlds used in early AI systems. It also excludes what one might call "spontaneous reactions": jumping out of the way of a moving vehicle, for example. These reactions are obvious decisions to extreme situations, and would also likely be excluded from any colloquial definition of everyday activity as well.[12]

There is an important consequence to the above definition: everyday activities are *relative to the agent performing them.* This is because the concept of an "everyday activity" is directly associated with the amount of knowledge (and in turn, the amount of experience) the agent has of the activity in question. To one agent, making tea may be an activity that has been experienced many times and learned from others; the actions involved in making tea are naturally constrained by the agent's knowledge. To another, making tea may be a foreign activity that is difficult and possibly dangerous. This relative nature is more of a cultural phenomena than an individual one. For example, I may be an expert at some particular activity (say, computer programming). This does not make computer programming an everyday activity to me, in spite of the fact that it occurs commonly. There are two reasons for this. First, despite the commonality of computer programming to me, there are many non-mundane aspects of the job: I often have to write things down and participate in the types of problem-solving activity that are not encompassed by everyday activity. I am also expected to perform at a much more than satisficing level: if professional programmers

---

12 In Chapters 3 and 4, the reader will see that while spontaneous reactions are not everyday activities as defined here, they are absolutely necessary to support more high-level activity. This is already fairly intuitive: it is useless to be able to plan to cross a busy street if one does not have the ability to jump out of the path of a moving vehicle if the situation should arise.

wrote all their programs at a purely satisficing level, none would hold their jobs for long[13]. Expert activities are fundamentally different from everyday activities, despite their common basis in the extensive use of knowledge. This difference will be explored further in Section 2.3.

2.2. Studying Everyday Activity

Having adopted a suitable definition of everyday activities, the nature of these activities may be examined. In part because of their mundane nature, but mainly because of their wide-ranging applicability, little practical work is available on the nature of everyday activities and the reasoning behind them. While the principles of problem-solving and the nature of such tasks have been relatively well-defined (e.g. [Polya, 1957; Cutler, 1985]), practical guidelines for everyday activity are rare. Because the domain is wide-ranging and the activities mundane, it is easy to make general statements about how everyday activities should be conducted, but difficult to apply them practically. For example, wise and revered ancients from Marcus Aurelias [1905] to Gracián [1992] have produced volumes of aphorisms on how to go about daily life. Since they are meant to be general, however, there are few instances of appropriate everyday behaviour that do not contradict at least one of the principles in any of these collections while obeying others. There is much that AI can learn from such compendiums, despite the fact that most modern equivalents are irreverent and anecdotal: for example, the simple principles of [Fulgham, 1986] have been shown to correspond to common characteristics of problems in the field of cooperative problem-solving [Durfee, 1992]. Making use of such knowledge in the form of programs, however, is also often much more complex than applying it in principle. Stefik [1981b] for example, notes the considerable gap between the advice on problem-solving tasks given by Polya [1957], and their realization in problem-solving programs, due to the amount of knowledge required to interpret this advice. For the aphorisms mentioned above, this gap is even more obvious, due to their abstract and often metaphoric nature as well as their generality.

Outside of these aphorisms, there is little in the way of recorded information on how to go about everyday activities. The reasons for this are simple. First, the knowledge we use to go about our everyday activities is so common that it is applied for the most part in an unconscious manner. This is also a common problem in expert systems research: an expert at a particular task uses his or her knowledge of that task so often and frequently that it becomes

---

13 Insert your favourite computer corporation joke here.

difficult for them to consciously piece together the reasoning processes involved [Waterman, 1986]. More significant than this, however, the knowledge involved in performing any everyday activity is highly distributed and usually learned very gradually: it is difficult to recall the knowledge surrounding a particular everyday activity as a collection, simply because it was not learned that way and is never referred to in that manner. Any everyday activity begins as something novel, and we become familiar with the activity (that is, acquire knowledge with which to guide its performance) only through many repeated performances. Thus the knowledge on which everyday activities is based is not consciously acquired: it is pieced together bit by bit over long periods of time. Because of these factors, most written work on everyday activities involves either an extremely specific focus on one particular facet of activity (e.g. on being environmentally conscious in everyday life [Cailliet et al., 1971]), or involves specific knowledge of aspects of a large set of activities, rather than very extensively describing a single activity. A good example of the latter is [Parker, 1886], a guide to household management. This work claims to be a "complete guide to household management", state of the art of its time, but is essentially a scrapbook of household tips and recipes[14]. It is useful only in specific situations and to one who already has significant experience with these activities.

The only time in the lives of most people in which there is extensive learning of everyday activities over a short period of time is in childhood. The one significant exception to this, however, provides another source of information about the knowledge involved in at least some everyday activities. As mentioned earlier, a great deal of the knowledge we use in performing everyday activities is culturally acquired: much knowledge for performing everyday activities is learned through others in society, who have in turn had this passed down to them. This includes both formal knowledge such as laws and rules, as well as informal hints that make activities easier to carry out. This knowledge, obviously, varies greatly from culture to culture, and forces one to make a vast number of adaptations in the way they perform their everyday activities when they make a geographical move from one culture to another. Although they are rare, guidebooks for new immigrants from other cultures are a useful source for information about the everyday activities in our culture. One such guide [EmpImm, 1991] consists of a large

---

[14] These recipes are vague and poorly described, leaving out vast amounts of general knowledge compared to modern recipes. Recipes are especially relevant to everyday activities, in that their sole role is to guide an ongoing course of activity at which the person performing the activity is assumed to have a collection of experience-based knowledge required to apply the recipe. The nature of this role is discussed in Section 2.3.

collection of useful facts about everyday activities in Canadian culture. This guide provides knowledge about such everyday activities as making transactions in a grocery store; dressing for winter; obtaining medical services; and what a typical job interview consists of. Many of the facts listed in the guide would be considered obvious by anyone who has lived in Canada for any length of time; for example:

- that arriving more than 10 to 15 minutes late for a business meeting or 15 to 30 minutes late for a social meeting is considered inappropriate;
- that waiting in line is common for many activities, and attempting to move ahead in a line is not considered to be socially acceptable;
- that merchandise is usually packaged in bags when sold in order to identify it as purchased when it is being removed from the store.

However, they are all important pieces of knowledge that aid us in going about our everyday activities. As can be seen from these examples, they are much more immediately applicable than the aphorisms discussed previously. However, these bare facts seem to say very little about how we go about *applying* this knowledge in performing everyday activities[15].

In contrast to these efforts, comparatively little of what could be termed formal analysis has been performed on the nature of everyday activity or the types of knowledge and processing mechanisms humans make use of to perform everyday activities so simply. Of that in existence, much has been performed from the perspective of psychology and sociology. One area that has concerned itself strongly with everyday human activity is environmental psychology[16], which studies human behaviour in the context of its setting. A number of notable attempts have been made to examine everyday behaviour from the perspectives of environmental psychology (e.g. [Scribner, 1984; Lave et al., 1984; Barker, 1968; Barker et al., 1978]). These attempts have, above all, demonstrated that the behaviour of a given agent is intimately related to the environment around it. The setting in which behaviour occurs constrains the behaviour of an agent, and similarly, the behaviour constrains and alters the settings in which it takes place [Barker et al., 1978; Barker, 1968].

---

15 In fact, I will argue that the *format* of this knowledge (as opposed to the knowledge itself) says a great deal about the way we apply that knowledge in the performance of everyday activities.

16 Also termed *ecological* psychology.

Much can also be learned through informal examination of episodes of everyday activity. In fact, one of the most important contributions of environmental psychology is that formal studies of behaviour often artificially bias the behaviour under study. For example, a psychologist may make up a questionnaire in order to examine a certain specific behaviour. By doing this, the psychologist is dividing behaviour into specific quantifiable parts in order to isolate the behaviour under study. However, at the same time the psychologist is also usually disregarding the existing structure of behaviour, which may not be obvious, and selecting the parts in which he or she is interested. This changes how the behaviour is viewed, and results in a biased view of behaviour as a whole [Barker, 1968; Barker et al., 1978][17]. Thus, informal examinations of behaviour play a unique part in examining the structure of behaviour as a whole.

In addition to the complex, routine, and knowledge-intensive nature of everyday activities inherent in everyday activity, there is much one can learn from the examination of episodes of everyday activity. Consider the following example[18]:

> Jane was in the kitchen, and wanted a cup of tea. She checked to see that the kettle on the stove had enough water in it. It did not, so she filled it with water, put it back on the stove, and turned the stove on. Then she got her favourite cup from the cupboard, took a tea bag from the tea canister, put it in the cup. The phone began to ring, so she walked over and answered it. It was Dick, calling to discuss plans for the weekend. Dick and Jane talked until the kettle whistle sounded. Jane said she had to go and bade Dick farewell. Then she poured the hot water on her tea, and waited a minute or so until the tea became dark. She added milk and sugar, and then went to call Dick on the telephone to finish their conversation.

Looking at this example, one could attempt to develop a list of the pieces of knowledge involved in accomplishing this activity, must as must have been done to develop [EmpImm, 1991]. However, for the purposes of understanding the cognitive mechanisms that support the performance of everyday activities, we must begin by examining the underlying phenomena demonstrated by the activities themselves. This example illustrates a great deal about everyday activities:

---

17 This is in fact a common problem in AI. Some instances of this problem will be demonstrated in the next Chapter.

18 The principles described in this Section apply to all the episodes of "everyday life" described in [Agre, 1988] and, I argue, to *any* episode of everyday activity. The reader is encouraged to apply these principles to any episode he or she cares to choose in order to demonstrate the obvious and common-sense nature of these. Those desiring more examples are directed to [Anderson, 1991], an early draft of many of these ideas.

- *Everyday activities overlap.* This is a fundamental property of everyday activities, and comes about because of their common nature. This overlap occurs in two ways. First, everyday activities subsume one another. For example, filling a kettle and boiling water can be termed everyday activities. If one had to search for sources of water and then try to figure out how things like taps worked, as well as the behaviour of water itself, it would be a complex task. Knowledge reduces it to a fairly simple one. However, the everyday activity of making tea subsumes the everyday activity of boiling water. In addition to this subsumption relationship, everyday activities interact. Answering the phone can be considered an everyday activity, and here the phone-answering activity occurred in the middle of the tea-making activity. Any significantly long episode of everyday activity will display both of these phenomena.

- *Everyday activity is ongoing.* This fact has been noted previously by Agre [1988] and Chapman [1990]. Everyday activities blend together in an ongoing manner: new activities arise and take the place of old ones, and because of the overlap described above, there is often not a clear transition from one activity to another. The subsumption relationship described above makes this especially difficult. For example, Jane may be making tea while camping (if Jane camps often enough for it to be mundane and common, this would be an everyday activity), making tea would be part and parcel of the overall camping activity, which would still be ongoing long after the tea-making had been completed. The tea-making activity would also likely give rise to other activities: drinking the tea, washing dishes, etc.

- *Everyday activities occur over wide time spans.* Making tea is an everyday activity, and takes a reasonably short time. Boiling water or filling a kettle are also everyday activities, and take even less time. Driving along a freeway system is also an everyday activity [Norman, 1988; Miller, 1994], and takes considerably longer. Time is not particularly relevant to differentiating between everyday and non-everyday activities. The time available to deliberate about activities also varies: some demand immediate responses (e.g. deciding whether to answer the phone), others within a few minutes (e.g. leaving a kettle to boil while on the phone), and others much longer periods.

- *Courses of action within one everyday activity will often contradict those of another with which that activity interacts* [Donagan, 1987]. In the interaction between tea-making and phone-answering, the telephone could just as easily have rung when the tea was darkening, or while adding milk and sugar: times when interaction would not have been as

conveniently dealt with as it was above. Reasoning in everyday actions somehow allows us to extract ourselves from most of these situations.

All of these characteristics are important; however, one of them, the ongoing nature of everyday activity, deserves further mention at this point. In traditional AI planning terminology, agents perform *tasks*. However, the use of this term in conjunction with everyday human behaviour is misleading: it tries to characterize as easily differentiated something that is continuous and highly interwoven. While in principle everyday activities consist of numerous tasks, the actions contributing toward their completion are highly interleaved: in many cases, it is impossible to completely isolate a given task. As mentioned earlier in this Section, attempting to completely (and artificially) isolate particular tasks for study has led to many misunderstandings about the nature of everyday activity. Viewing everyday activities as tasks also contributes to a misunderstanding of the nature of everyday activities, advocating a view of such activities as hurdles that must be overcome or problems that must be solved, rather than routine situations in which an agent participates in an ongoing manner. The term *task* is well-suited to the types of problem-solving activities AI is used to dealing with (Section 2.0), but is poorly suited to most of the everyday interactions humans have with the world around them. As Agre and Chapman [1989] put it, embodied agents *lead lives*; they don't *solve problems*. For these reasons, the term *activity* is used throughout this dissertation, as opposed to the term *task*. The former term subsumes the latter: making tea is an activity that coexists and interacts with a number of other activities in which an agent is participating, rather than a task to be dealt with or a problem to be overcome.

The ongoing nature of activities also extends past the point of tasks. Actions, like activities, subsume and interact with one another. Not only does an everyday activity such as making tea encompass other activities such as filling kettles, but these also contain more fine-grained activities. At some point, AI planning systems normally denote atomic units called *actions* that are assumed to be directly executable[19]: picking up the kettle, for example. However, even these are grouped collections of smaller definable movements, such as moving individual fingers to grasp the kettle. Even below this level, sets of muscles must be coordinated together to accomplish desired movements. Even at this level, overlap is considerable: when speaking, for example, the configuration of muscles involved in producing a

---

19 The details of the operation of such systems will be dealt with in Chapter 3.

particular sound depends heavily on the sounds that will shortly follow in the speech stream [Wright, 1990].

The relevant aspect of this from the point of view of cognition in everyday activity is where the conscious aspect of this hierarchy of activity begins. It seems unrealistic (and tedious) to reason about individual muscle interactions. On the other hand, it seems unacceptable that no cognitive activity goes on below the level of complete tea-making either. It is postulated by many that this low-level coordination is handled by *motor programs* [Wright, 1990] that control individual muscle movements and are invoked by more higher-level cognitive functions. Where the level begins that we can actually begin to assume that the conscious control of actions begins is difficult to determine. However, a generally accepted boundary is approximately 100 milliseconds [Wright, 1990; Hofstadter, 1985]. This is regarded to be the point where conscious control ends and "subcognition" [Hofstadter, 1985] begins. The research presented in this dissertation assumes that such a level exists (although existence at the 100 ms level is not of great importance), and that actions can exist and be reasoned about as symbolic structures above that level.

2.3. Cooking as an Example of Everyday and Expert Activity

> Apparently it can require more to be a novice than to be an expert, because (sometimes, anyway) the things an expert needs to know can be quite few and simple, however difficult they may be to discover and learn in the first place.
>
> – Marvin Minsky, *Why People Think Computers Can't*

> One need not be a professional chef to prepare gratifying food, or for that matter, to write about it.
>
> – Bob Blumer, *The Surreal Gourmet: Real Food for Pretend Chefs*

The examples of everyday activities described to this point have illustrated a great deal about the nature and characteristics of everyday activities in general. However, they have thus far said little about the cognitive mechanisms we employ when performing everyday activities or the structuring of the knowledge on which these performances are based. Little has also been done to this point in emphasizing the difference between everyday and expert activities with respect to these mechanisms, aside from citing differences in expectation of performance. In order to examine these phenomena, a more detailed examination of a specific domain must be performed. The domain of cooking was selected for this examination for two reasons. The first is its ubiquitous nature: everyone cooks, though some do it better than others. Cooking fits the definition of an everyday activity given in Section 2.1, and is also a good example of what most people would intuitively

call an everyday activity. From an AI perspective, cooking is a useful choice of domain because it has been employed previously in AI systems, thus facilitating comparison of techniques and examples employed by other research projects (e.g. [Agre, 1988; Hammond, 1989a; Hammond et al., 1990]). In addition to this, cooking is also is a domain that is commonly viewed as both an everyday activity to which the average person can relate, and also an intricate and involved activity when performed by experts. It thus affords a good vantage point from which to contrast everyday and expert reasoning.

2.3.1. Everyday Cooking and the Use of Recipes

Recipes are inarguably the cornerstone of cooking. We often think of recipes in written form, as one might see in a cookbook. However, just as one can describe from memory how we perform any everyday activity, one can describe how one prepares a commonly-made dish from memory. The professional chef commits many recipes to memory, and as mentioned in the previous Section, often uses those internalized recipes to create new dishes. However, the everyday cook also makes use of internalized recipes. My own such internalized recipe for making tea appears in Figure 2-3. Asking others the process by which they make tea would result in a structure similar in nature, but different in detail (e.g. someone else might commonly employ an electric kettle, or use loose tea).

Internal recipes such as that shown in Figure 2-3 form an important part of everyday cooking: for any dish that one has extensive experience at cooking, I argue, one can describe the basic preparation of that dish, resulting in a structure much like the one shown in the Figure. However, written recipes are used much more often by the everyday cook, and are usually of a much different structure.

1. Find a kettle and a teacup.
2. Fill the kettle with water, put it on the stove, turn the stove on.
3. Get a tea bag, put it in the teacup.
4. When the water is boiling, pour it over the tea bag.
5. Wait until its reasonably strong, and remove the tea bag.
6. Put some milk in the tea, and drink it.

Figure 2-3. General recipe for making tea.

Finding a written recipe for tea to compare to Figure 2-3 is difficult, since tea-making is such a common act that most of us commit it to memory long before reaching the stage where we would be expected to cook on our own. Such recipes are usually only found for specialized circumstances. For example, Figure 2-4 illustrates a recipe for tea for 100. Adapting a tea recipe to serve 100 is not an everyday activity for the majority of people; however, following such a written recipe is certainly an everyday activity for many of us. Most written recipes are more complex than that shown in Figure 2-4, but display a similar structure. Internal and written recipes are similar at a basic level: they are both resources of knowledge that can be applied to a task (in this case, cooking) to make it relatively simple. Written recipes are cultural artifacts, and have evolved in much the same way as tools for cooking (Section 2.3.2). Internal recipes represent a combination of previous experience, memorization of written recipes, and other cultural teachings.

4 1/2 Gallons water — 1 1/2- 2 cups tea

Fresh cold water brought to a boil to which add the tea, in cheese cloth bag. Remove from fire and let steep a few minutes, not more than three minutes. Remove the tea bag, and do not let the infusion boil.

Figure 2-4. Recipe for tea, 100 servings [Hadwen, 1937].

There is much that both internal and written recipes illustrate about performing everyday cooking and everyday activities in general. First of all, while my internal recipe for tea was described in steps, these steps need not be carried out in the exact order in which they are described. Other orderings work equally well, and when I use my internal description, I use whatever order seems convenient to my circumstances. I can also react to opportune situations (e.g. I wouldn't boil water if there was some obviously available), and make do with less-than-opportune situations (e.g. if there isn't enough milk, I could have whatever little there is my tea). More importantly, the internal recipe is neither completely specific nor completely general. It does not contain the individual actions necessary to make a cup of tea[20]. For example, even simple actions such as pouring hot water over the tea, or

---

20 Indeed, the reader will see in Chapter 4 that making tea involves a much more extensive series of operations than one might at first think.

removing the tea bag will differ at least slightly from instance to instance. Indeed, if the internal recipe did contain each specific action necessary, it would be useful only in one precise situation. The recipe also omits many absolutely necessary actions (e.g. turning off the stove). Somehow, an everyday chef has the ability to apply internal recipes as guides, making use of context separately to accomplish each of the steps involved - adding further activity as necessary and not even considering aspects of the general recipe that do not make sense in the context in which the agent finds itself. This is equally true for all everyday activities.

On the other hand, the internal tea recipe does contain *some* context. Not only does the internal recipe contain much more context than the written recipe in Figure 2-4, it contains some very specific contextual aspects: precisely those on which the agent usually relies when performing the activity. In this case, my internal tea recipe is assuming I am making only one cup of tea, that I am using tea bags, and that a stove is used to heat water: precisely the conditions that hold in the environment in which I most commonly make tea. In a different context, aspects would be described differently. For example, if I were specifically asked how I make tea at my office, I would describe the same general process using an electric kettle. This is an *entirely new* everyday activity: I do not consciously take the internal recipe from Figure 2-3 to substitute the use an electric kettle; rather, I am reminded of a different routine by a different set of external circumstances. However, knowledge about these separate everyday activities cannot exist in isolation, the way one would collect recipes in a cookbook: they are related to one another through the extensive knowledge required to actually perform each activity given these vague descriptions. In this case, much of the knowledge involved would overlap between making tea at my office and at home: some aspects of my tea-making knowledge are distributed and connected with given circumstances; other aspects are common and more unified. This implies that internal recipes (be they for cooking or for guiding other everyday activities) are not stored as program or plan-like structure, but in a more general, more accessible and flexible form. *They are by no means the algorithmic structures that they are commonly thought of.*

Written recipes, though seemingly more structured than internal recipes, show these same characteristics. The written recipe of Figure 2-4 is shown as a series of steps, and indeed, that is what recipes are commonly thought of. However, there is also a great deal of knowledge in any written recipe that is not in this form. For example, any written recipe has associated with it a large number of constraints, which express restrictions on objects or techniques. In Figure 2-4 for example, *steep no longer than three minutes* is a constraint on the behaviour of tea-making, as is using fresh cold water (as opposed to water

in some other condition)[21]. In fact, prior to giving the recipe shown in Figure 2-4, Hadwen [1937] states approximately a page of additional constraints, from not using metal containers for making tea to increasing the amount of tea rather than the time of steeping if a stronger flavour is desired. Such constraints are also an important aspect of internal recipes, but are not often thought of when attempting to list an internal recipe, since many arise for consideration only when we begin to carry the activity out. Other knowledge common to written recipes describes the condition of the ingredients and recommended tools. Although recipes have traditionally been thought of as plan-like structures, there is clearly much more to the average recipe than the steps for its successful completion.

In addition to omitting much contextual detail, there is also a great deal knowledge in other forms that is omitted from any written recipe simply because it is understood to be applied automatically[22]. A step in a recipe may call for chopped vegetables, for example, without describing how this process is to be performed. This "understood" knowledge can be of many types. For example, words like *add*, *mix*, or *stir* are used commonly, assuming we know the ingredient-specific differences between these actions; common features of ingredients are omitted (e.g. no recipe ever states to remove eggshells from eggs); and variable elements (where to go in the kitchen, often what tools to use) are omitted because they would simply be too specific to make sense of in most circumstances. Here again, written recipes are by no means algorithmic structures.

The amount and type of "understood" information in a recipe has a great deal to do with its audience: a recipe for making tea will usually be as unstructured as the one above, and may have a great deal of understood information, while a recipe for a more exotic dish (from the perspective of the cook: a foreign dish is a good example) would likely be more detailed. Proper recipe-writing standards for written recipes minimize the amount of implicit information. Using standard terms for cooking processes, for example, make them more easily followed by everyday cooks [Peckham and Freeland-Graves, 1979; Hullah, 1984]. In addition to the understood information present in recipes, there is understood information about following a recipe itself: that

---

21 The necessity of such constraints can also be seen in any recipe-like structure, such as the brief guides to chores such as ironing and washing described in [Parker, 1886].

22 This is typically called *common sense* knowledge in AI. Again, it is also an important aspect of internal recipes, and is not thought of when recollecting an internal recipe, simply because it is only considered when needed, like the constraints described above, and is separate from the recipe itself.

the ingredients usually appear at the top; that the form of ingredients is to be that encountered normally unless told otherwise; that special suggestions may appear at the bottom of the recipe. As mentioned above, following a recipe is itself an everyday activity, one that is part and parcel of the larger everyday activity of cooking. Each clearly contributes some contextual information that guides the process along (again, in the manner shown in Figure 2-2).

This understood information is what really separates an internal recipe from a written recipe. While they may look fairly similar once an internal recipe is written down (e.g. Figure 2-3), they are different in that an internal recipe is highly connected with a great deal of experiential knowledge that is present but consciously omitted when writing such an internal recipe down. This is precisely the understood information a written recipe relies on. In order to understand a written recipe, we must first read it and integrate it with this knowledge; with an internal recipe, this integration has already been already performed. In writing down an internal recipe, we also make a great many assumptions about circumstances in which the everyday activity will be carried out. For example, in Figure 2-3, I assumed that the majority of contextual factors were exactly as they are found in a typical instance of my making tea at home. Such assumptions allow us to put down in linear form what is essentially a non-linear structure. By non-linear, I mean that the internal recipe does not exist entirely in the sequential form that is shown in Figure 2-3. As mentioned previously, there are both general and context-driven aspects to these internal recipes. In writing down the internal recipe, we are recalling it in an artificially linear form by making a set of assumptions that allow a complete general sequence of activity to be written. That is, these assumptions provide an artificial context that allow us to select from the distributed aspects of this knowledge. Thus, if my internal recipe in Figure 2-3 had contained a step to turn the stove on and another specific one to turn it off, and I were specifically given a kettle to make tea with instead, there would be no conscious search of the sequence in order to find aspects that are incompatible with using the kettle (e.g. finding and removing the "turn off stove" activity). Rather, this aspect would never arise, since we are not using a stove: it was only included in the original sequence by virtue of the fact that a stove was assumed. Somehow, these internal recipes must exist in a format that supports this non-linear structure[23].

---

23 The hierarchical, subsumptive nature of everyday activities also supports this idea.

While much additional knowledge is required to apply an average recipe (internal or written), recipes are much more obviously viewed as suppliers of information: Recipes are used as *guides for activity*. As recipes are not algorithmic structures, they cannot be followed as such. Indeed, even average cooks often do not follow written recipes in minute detail [Peckham and Freeland-Graves, 1979]. However, recipes are crucial to everyday cooking: as caches of previous experience, they constrain one's actions in the manner shown in Figure 2-2, and provide the knowledge necessary to minimize danger and uncertainty in cooking. Recipes, in the words of McGee [1984], "are meant to help us prepare food...*without the distraction of having to think*"[24]. The concept of minimizing intellectual effort is important not only in common cooking activities, but in any everyday activity. As mentioned above, there is no conscious adaptation of an internal recipe from one situation in which it is commonly used to another[25], nor is there any conscious adaptation of a completely general (context-free) tea-making recipe to a specific situation with which the agent is familiar. The intellectual ease with which everyday activities are performed dictates this: such adaptations are computationally complex, and relying on such mechanisms in any complex domain would result in such a slow response as to make many everyday activities unperformable. On the other hand, adaptation is clearly required when the situation is less than familiar, and there will be a degree of unfamiliarity in any situation, no matter how common the activity. These require intellectual effort to the degree that the agent must go beyond the detailed, often-applied knowledge it has of the activity (i.e. the internal recipe). Some adaptations require very little intellectual work, such as when taking care of small but unavoidable differences between the recipe and the world the agent inhabits. For example, the exact spot one puts a cup down in order to pour tea into it will always differ from instance to instance. More intellectual work - the application of deeper domain knowledge - is required in other cases. For example, if I am at home making tea and the stove breaks down, I will expend intellectual effort searching for an alternative method of heating water (e.g. using an electric kettle, which is not normally part of that routine). The scale of such adaptations is broad, and can go far beyond what is considered an everyday activity, as will be seen in the following Section.

Intellectual work is most definitely required in everyday activities; the secret to their ease and timely performance is that the intellectual work is not

---

24 Emphasis mine.

25 Though of course such adaptation is possible and occurs in more complicated (and comparatively much more rare) circumstances.

wasted where it is not needed. By making use of the experience represented in internal recipes such as the tea-making recipe of Figure 2-3 as surface knowledge, saving the time spent searching for the best way to perform the activity, and using the deep knowledge that lies behind such recipes to perform adaptations such as the one above, we combine the timely responses, ease of performance, and adaptive behaviour that characterize our own everyday activities.

2.3.2. Comparing Everyday and Expert Cooking

Despite the emphasis thus far on its everyday aspects, cooking is obviously not limited to the amateur. Indeed, the term "everyday chef" used in the previous Section sounds unusual when it is first encountered, since our image of a chef is not usually of an ordinary person in a kitchen cooking hamburger, but one in gleaming white clothes and a tall hat concocting exotic dishes. On close examination, there are also many differences between the reasoning performed during cooking by an expert as opposed to an amateur.

From the perspective of an amateur, there is a great deal of knowledge employed by the professional chef that is simply not needed for preparing a passable meal [McGee, 1984]. Complex chemical and (in some cases) biological processes are at work when one prepares a meal [Seelig, 1991, Coenders, 1992], and in most cases the average cook is unaware of their operation, except at an extremely high level. Consider, for example, the use of baking powder. As Seelig [1991] explains, "double-acting" baking powder is essentially baking soda combined with two acids. The first acid reacts with the baking soda in the presence of water at room temperature, releasing carbon dioxide bubbles. These bubbles allow batter to rise. However, cake batters are quite thin, and often the gas bubbles escape before the batter has cooked and thickened enough to trap them. The second acid reacts only at higher temperatures, and thus produces more carbon dioxide when the batter has thickened enough to hold them. Now, very few average people would have enough knowledge of chemical properties of baking soda to provide such an explanation. They simply make use of the fact that putting baking powder in batter causes it to rise. To the professional chef, however, deep knowledge about this and many other common items is crucial, since in commercial use, successful recipes must be guaranteed (one of the basic differences between everyday and expert activities). The most basic knowledge of baking soda is suitable for everyday cooking activities; however, when limited to this knowledge along, the activity can break down in situations with which the everyday cook is not experienced. If the everyday cook were to try to construct a novel recipe, for example, the acid balance in the batter might be such that the baking powder reacts too quickly, causing the cake to fall. Expert knowledge is required for

this activity. By the same token, however, constructing a recipe is not an everyday activity to the average person: it involves many kinds of deep reasoning about many ingredients, interactions between them, and the processes for combining them. The average person does not have such knowledge simply because it is not usually required.

Tools for cooking provide a similar view of the contrast between expert and everyday behaviour. Anyone who has entered a department store or watched a cooking show on television knows that there is a large number of cooking gadgets on the market today, but few realize how large this number is: Campbell [1980], for example, illustrates more than two thousand different cooking tools and special techniques for using them. Tools are examples of knowledge in the world that has evolved through culture [Agre and Horswill, 1992]. Individuals modify existing tools over generations to perform a given task better or to apply to an entirely novel task, and we simply use these tools without appreciation of the knowledge behind them. Agre and Horswill [1992] argue that a great deal of knowledge in cooking is provided simply through the existence of specialized tools for specialized cooking tasks (pastry cutters for pastry, spatulas for turning, etc.). In contrast to what might be expected from this, however, most expert chefs do not make use of many of these specialized tools [Campbell, 1980]: they use fewer and more general types of cooking tools. A reasonable explanation for this is that expert "knowledge in the head" replaces the "knowledge in the world" represented by tools. Mastering techniques implies not needing assistance in the form of specialized tools for performing those techniques, and thus an expert chef can make use of more general tools in specific ways. Watching an expert chef illustrates this: we have all seen chefs separate eggs using an eggshell rather than a specialized tool; smashing garlic with the side of a knife; or making a great show of quickly and finely chopping vegetables without the aid of a food processor. I would argue that even the amateur does not make use of tools as world knowledge as extensively as Agre and Horswill [1992] presume. When I need a tool to stir a liquid, I choose a large spoon over a vegetable spoon, not because the latter is meant to be used for vegetables, but because it has holes in it that do not lend themselves well to stirring. Informal reasoning about function, often simply by looking at a tool, seems to play a much greater part than Agre and Horswill give credit. Most of us have a great drawer full of kitchen gadgets, and as often as not reach in

looking for something that will do an adequate job for the task at hand, rather than assuming there is a specific tool for every job[26].

While there is clearly a difference in the level of knowledge used by expert and amateur chefs, there is also a difference in the types of activities they engage in while cooking. Experienced chefs employ the types of knowledge discussed above to design and test new recipes. As mentioned above, they require intimate knowledge of the role of various substances and the manner in which they interact with one another. A system known as *Chef* developed by Hammond [1989a] demonstrates this type of reasoning. The system makes use of its previous experience at cooking (testing recipes in simulation and remembering and learning from its failures) and deep knowledge of the ingredients and processes involved in order to adapt old recipes to fit new situations (e.g. altering a beef-and-broccoli stir-fry dish to make an appropriately-flavoured chicken dish [Hammond 1989a]). This is no doubt expert reasoning, but the professional chef is also called upon to perform reasoning that is much more sophisticated than that implemented in Hammond's system. Hullah [1984] illustrates many complex examples, such as adapting recipes for microwave or convection cooking, or for lower fat, sugar, or sodium. In viewing the guides for performing such adaptations [Hullah, 1984], one sees that much of the knowledge involved is similar in structure to that used by many expert systems applications.

In contrast to this, the average cook does not have much of the knowledge that an expert chef possesses about ingredients and processes. The everyday tasks performed by the average person are much more mundane than the professional recipe adaptations described above: everyday cooking simply does not require such knowledge. An average person performing cooking as an everyday activity uses very high-level and limited knowledge about ingredients and processes to perform the very slight adaptations necessary to carry out simple recipes in the average kitchen. In most cases, the results are satisfactory, although not approaching the performance of an expert chef. More significantly, when problems arise (e.g. bread falling), the deep knowledge and reasoning processes of an expert chef allow a much better chance of recovery than those of an average cook. The non-expert simply lives with the possibility of failure: part of the definition of everyday activity, and an alternative usually unavailable to the expert.

---

26 Or more often than not, look for the traditional tool for doing something and, eventually tiring of sifting through dozens of gadgets cluttering the drawer, reach for something that looks like it is remotely applicable.

The knowledge and reasoning mechanisms used by amateurs and experts in cooking say a great deal about how expert tasks and the reasoning behind them relate to everyday activities. Expert performance depends critically on expert knowledge [Hayes-Roth et al., 1983]: an expert employs specialized knowledge and reasoning mechanisms to perform at a high level in a narrow domain. Thus in order to perform the complex recipe adaptations described above, a professional chef knows about the biochemical composition of ingredients, the manner in which they interact with one another, and also has complex knowledge on how to control the complex processes that occur when they are cooked. Just as critically, however, everyday activities depend critically on general knowledge, and we apply this general knowledge in order to perform adequately in a wide range of domains. In order to perform everyday cooking activities, we possess knowledge of how to follow recipes, and fairly limited knowledge of ingredients and how they interact. We fail in some situations, because expert knowledge is sometimes required in complex situations, but we live with the possibility of failure. Everyday life is complex, and dealing with it does not mean getting rid of the complexity, but performing adequately within it [Davis, 1991][27].

2.3.3. Further Characteristics of Everyday Activities

Overall, we see that an examination of cooking as an everyday activity reinforces many of the points that have already been made in this Chapter. Cooking can be complex, but in situations where one has experience in performing many of the activities involved, cooking is easy: simple, standard recipes become trivial; some that occur very often, such as making tea, are not even thought of as recipes any more, and are performed with little decision-making activity. However, this examination has also demonstrated new phenomena about everyday activities from the point of view of the cognitive mechanisms necessary for performing them:

- *Everyday activities can be easily described from memory.* We can take any activity that we are very familiar with and describe it from memory in a manner similar to the internal tea recipe of Figure 2-3. Moreover, I

---

27 This may seem an obvious fact, but is often overlooked by those in computer science obsessed with complexity analysis. Virtually any significant domain in AI involves intractable problems; however, humans tackle many such problems (in particular, those relating to everyday activities) with ease. Complexity theory is often misleading in problems such as these, because its results are based on its design of pathological instances, rather than the many instances that can be tackled with good results using computationally feasible rules [Goldberg and Pohl, 1981].

argue that the amount of detail shown in Figure 2-3 is typical of a general description of any everyday activity. Far from being context-free, it contains aspects of context that are especially common.

- *These memory structures are more than just sequences of steps.* While the sequence of steps necessary to perform an activity is obviously important, much of the experience we have with an activity is in other forms: recommendations of particular tools, or restrictions on objects or techniques, for example.

- *These memory structures represent what typically goes on in the course of an everyday activity, but are not detailed enough to control it directly.* Even a simple action like making tea cannot be completely programmable, because no situation will exactly match the typical scenario for any everyday activity. Subtle variation within the activity and interaction between tea-making and other activities require us to diverge from the typical or normal methods with which we have the most experience in order to adapt to ongoing changes in circumstance. This requires deeper information about the task at hand. While we can describe how we go about the activity in general, we usually omit from these descriptions much of the information necessary to be able to deviate from this general methodology. Clearly however, this deep knowledge must exist. Because we are dealing with everyday activities, most adaptations should by definition be minimal, and these deep forms of reasoning are used as little as necessary. However, they are nonetheless essential.

- *Intellectual effort is minimized during the course of everyday activities.* Just as recipes allow us to prepare food without the distraction of having to think, the internal recipes (i.e. our previous experience) that we follow for everyday activities allow us the same convenience. There is little in the way of complex adaptation in any everyday activity, and (as described earlier) each step an agent takes during the course of an everyday activity should seem obvious: that is precisely why we can describe an everyday activity from memory. Indeed, in many cases we often only think about what we are doing when something goes wrong. For example, when making tea in my home, I don't generally reason that I turn the stove on specifically because I am heating water. It is simply part of my routine. If I stop and analyze what I am doing, these "higher purposes" can be easily considered. This might happen, for example, if the stove were broken. However, they are usually not thought of or reasoned about *automatically* in the course of an everyday activity.

- *We may have more than one way of performing a common activity in different situations.* These methods are commonly termed *routines* [Agre, 1988; Agre and Chapman, 1990]. One routine will usually seem obvious within a particular situation [Agre, 1988]. Each effectively represents a different everyday activity.

- *Knowledge about a particular everyday activity does not exist in isolation.* If I have two common routines for making two everyday items (e.g. making tea and coffee), the routines are not completely separate from one another. While each will involve different basic steps, they will often share a great deal of background knowledge. For example, both making tea and making coffee have the heating of water, the use of cups, and a great deal of general kitchen knowledge in common, despite being possibly very different routines.

2.4. On Rationality and Everyday Activities

> **Rational**, adj. Devoid of all delusions save those of observation, experience and reflection.
>
> – Ambrose Bierce, *The Devil's Dictionary*

Another concept crucial to studying everyday activity and the reasoning mechanisms behind it is that of *Rationality*. The general concept of bounded rationality, that appropriate decisions must be made in light of limited knowledge and deliberation time, has already been mentioned. However, the nature of rationality itself has yet to be discussed. Clearly, rationality is an important characteristic in intelligent systems: no AI researcher would claim to want to create irrational programs. But what is rationality, and how can rational actions be distinguished from irrational ones? In general, rational behaviour can be defined as "doing the right thing" at each moment [Agre, 1988; Chapman, 1990; Russell and Wefald, 1991]. But what is the "right thing"? In the past, the right thing has often been defined from a decision-theoretic perspective (e.g. [Simon, 1957]). From this perspective, a rational agent is one that attempts to mathematically optimize benefit, utility, or comfort within the bounds of its resources [Norman, 1988]. Thus, the right thing is the alternative with the highest-weighted combination of factors in its favour after having considered it for some particular length of time.

For problem-solving that involves extensive deliberation, where the factors that make alternatives worthwhile can be measured and combined numerically, the decision-theoretic perspective is extremely useful. However, as has been discussed extensively throughout this Chapter, everyday activity is fundamentally different from this type of problem-solving. Everyday

activity does not lend itself well to the decision-theoretic approach because of its eclectic and context-sensitive nature, and because the factors that make an action optimal are often subtle and difficult to accurately express numerically. In addition to these practical limitations, there is also a philosophical incompatibility between the mundane and common nature of everyday activities and the concept of intricately weighing alternatives against one another, counting up pros and cons. Even strong proponents of the decision-theoretic perspective point out that there are many situations where quantitative estimates of the attractiveness of an alternative are unavailable, and that rational analysis can still be carried out in spite of the absence of qualitative features [Simon 1969]. Indeed, attempting to precisely define and quantifying all aspects of a decision may in fact be detrimental to the decision-making process. In management science, for example, the view that decision-making should be a formal, precisely-defined process has arisen largely from studying low- and middle-level management decisions, yet is applied mainly to top-level decisions. The latter are largely ill-defined and unquantifiable, and the application of decision theory to these problems is more likely to cause organizations to pursue inappropriate courses of action more efficiently [Minzberg et al., 1976].

For everyday activities, moving beyond completely quantified, decision-theoretic rationality is crucial: as has been pointed out by Barrett [1962], universally-applied rationality of the decision-theoretic variety would be indistinguishable from psychosis. In contrast to the decision-theoretic perspective, any action taken by an agent participating in an everyday activity is usually an obvious one: ridiculous or irrational alternatives are not weighed and abandoned - they are not consciously considered in the first place. Rationality in everyday activity is defined by culture, experience, and precedent. The rational alternative - the "right thing" - is one that an agent adopts with the full expectation that it will lead to some goal in light of the agent's knowledge of the activities in which it is participating or intending to participate. That is, the rationality of an action lies in the context of the particular, concrete circumstances in which it is chosen [Suchman, 1987]. Failure to consider this body of knowledge results in an inadequate concept of rationality. Newell's [1988] *principle of rationality*, for example, simply states that actions are selected to attain the agent's goals. Failure to include the concept of goals conflicting with one another and with the general contextual knowledge surrounding the agent's activities allows this definition to include many activities that are objectively irrational. For example, If I were making tea, and the house started on fire, it would be perfectly rational within the bounds of this definition to stay in the burning house and accomplish the tea-making goal.

Choosing the rational alternative is thus the product of applying appropriate contextual knowledge to resolve uncertainty. This knowledge naturally constrains an agent's choices, in the manner depicted in Figure 2-2: it is what makes most actions in everyday activity obvious. This view is completely encompassed by the concept of bounded rationality: there are simply too many alternatives to rely on enumeration and weighting. Very little in the way of heavy deliberation characteristic of the decision-theoretic model is involved in everyday activity. Rationality does not mean performing the "logical" action in the formal sense of the word; it means doing something that is compatible with the agent's past experience, future intentions, and knowledge of the situation in which the agent finds itself. This form of rationality is very different from the decision-theoretic perspective, but I argue keeps much of the spirit decision-theoretic rationality encompasses.

This view of everyday rationality has some implications not encompassed by decision-theoretic rationality, however. By basing rationality on experience and knowledge of the situation in which the agent finds itself, the rational decision is not necessarily *objectively* rational, but rather rational to the extent that it reflects what the agent knows. Thus many decisions that are rational to one agent may seem irrational from another's perspective, or even in retrospect from the agent's future point of view.

Unlike others (e.g. [Agre, 1988; Chapman, 1990]), I do not believe that the fact that the most appropriate course of action is usually obvious when performing an everyday activity implies that agents can be organized as automatons, where every decision is predetermined. The difference between this view and that of Agre and Chapman is that rationality involves the appropriate run-time application of contextual knowledge, as opposed to making a pre-programmed response to a given situation[28]. The context affecting an agent's decisions is much more complex than is necessary for simple stimulus-response reactions to be responsible for everyday activities. This context is not simply the product of the state of the environment around it, which can be recognized and responded to in a primitive, behaviourist manner. Rather, it is the product of both the environment around the agent, the internal state of the agent itself in the form of goals and desires, and complex interactions between the two.

---

28 This view of coherent activity as emergent from many pre-programmed responses will be described in detail in Chapter 3. It is mentioned here only to make distinct the differences in the basic view of everyday activity it entails.

From the perspective of rationality, making appropriate use of this context is much more complex than I argue can be accomplished with simple reactions. Even though knowledge obtained through previous experience may immediately eliminate many inappropriate alternatives, this contextual knowledge may not all be immediately available. For example, the agent may be participating in more than one activity at a time, and it make take some deliberation to realize the interactions between them. Likewise, an agent may intend to participate in a great many activities in the future, and the interaction of these with current activities may not be immediately obvious. More general contextual knowledge may also be required to realize the advantages of one alternative over another, and may not be immediately available.

These aspects of the contextual knowledge provided by experience can be defined in terms of *compatibility* and *coherence*. Any action in everyday activity that is carried out by a rational agent needs to be be:

a) *Compatible with the environment:* the action should be an appropriate response to the situation in which the agent finds itself.

b) *Locally coherent:* the action selected should be as compatible as possible with other activities the agent may currently be pursuing. Since, by the very nature of everyday activities, an agent is often participating in many activities at once, there are many ways in which carrying out a single action working toward one goal can invalidate another.

c) *Globally coherent:* the action should be as compatible as possible with the activities an agent intends to perform in the future. For example, the agent may be about to spend all of its money purchasing some article, and yet possess the intention to go out to dine in the near future. Clearly, carrying out the first action will invalidate the second, assuming that the dining intention is the more important of the two.

Clearly, these aspects are in competition with one another, and sacrifices must often be made as bounded rationality dictates. I argue, however, that when sacrifices are made, they are made by virtue of the agent's own extensive knowledge of the activities in which it is participating. As will be shown in Chapters 4 and 5, the difference between this perspective and that of Agre [1988] and Chapman [1990] is one of appropriate bounded reasoning, allowing the various aspects of context to be considered as they arise (and as time allows), as opposed to complete compilation. In any significant situation, I argue, the complexity of the contextual knowledge precludes complete

compilation, and the complexity of the interactions between everyday activities preclude the strict use of such pre-compiled structures.

2.5. Summary

This Chapter has defined everyday activities and differentiated them from the expert activities more commonly associated with AI systems. It has also sought to characterize everyday activities, through both high-level examinations of types of everyday activity and detailed examinations of a particular domain. The characteristics of everyday activities discussed in this Chapter are summarized in Figure 2-5. As should be obvious by this point, these are by no means independent of one another. In fact, a common theme arises from them: *everyday activities are complex and uncertain, and the knowledge possessed by the agent performing the task must be applied to resolve that uncertainty. The correct application of the extensive knowledge that is a part of all everyday activities by definition is what makes everyday activities simple for humans to perform.*

Applying this knowledge, however, is far from simple. Many of the characteristics described in this Chapter centre around the organization of an agent's experience: in general, that an agent performing an everyday activity makes use of a routine that is well-suited to a general set of circumstances but not completely specific. Such an agent must also have an extensive collection of knowledge about the world around it, and the ability to apply that knowledge to allow it to perform the routine flexibly in a specific situation. This knowledge also allows the agent to integrate the recommendations of its experience at one activity with that of other activities in which it is participating. Such an agent must be goal-directed, in that it must make use of its previous experience with a commonly-performed activity to work as directly as possible toward its objectives. At the same time, the agent must also be adaptive, in that its knowledge of the activity cannot serve as a complete guide in any particular set of circumstances and must be altered as circumstances change as well as integrated with other activities. These are precisely the characteristics that Sternberg and Salter [1982] argue are the basis for all intelligent behaviour.

Everyday activities overlap (subsumption and interaction)
Involve the use of large quantities of knowledge (in the head and the world)
Everyday activities are mundane
Everyday activities are common
Everyday activities are ongoing
Everyday activities occur on a wide time scale
Dealing with everyday activities involves contradictions
Characterized by adequate as opposed to expert performance
General operations of everyday activities can be easily recalled from memory
Experience with everyday activities involves minimizing intellectual work
Everyday activities are recalled as a mixture of general and specific information
Knowledge about a particular everyday activity does not exist in isolation from that of other everyday activities
This information is non-linear in organization and is much more than just a sequence of steps
Concern for current activities must be balanced with concern for intended activities in the future

Figure 2-5. Summary of the characteristics of everyday activities.

From the characteristics discussed in this Chapter, it is immediately obvious that performing everyday activities demands a great deal from the agent's cognitive machinery; much more than performing expert activities. The organization of experience that characterize everyday activities demands flexibility in representation, while the combination of goal-directed and adaptive behaviour requires that this knowledge be applied in an extremely flexible manner.

With these characteristics in mind, Chapter 3 reviews past techniques in AI planning systems in terms of their ability to serve as the basis for a cognitive architecture for everyday activities.

CHAPTER 3

AI PLANNING MODELS AND EVERYDAY ACTIVITIES

> The crucial element of all scientific models is change. They must evolve as we learn more about the world. Sometimes the change is no more than tinkering with a small part of a model. At other times, as when the Copernican model took over from the Ptolemaic one, a real revolution occurs.
>
> – Zeilig and Gaustad, *Astronomy*

3.0. Introduction

Having defined everyday activities and differentiated them from other types of activity, it still remains to be seen how the reasoning behind everyday activities can be captured and implemented computationally. As has been previously mentioned, reasoning about action has always been a topic of interest to AI, and there are many perspectives from which to view reasoning about action. This Chapter provides an overview of models for reasoning about action previously used in AI systems, and argues that none is suitable for everyday activities as defined in Chapter 2. This overview is not intended to discredit or belittle these models, but rather to view their performance both objectively and from the perspective of their application to everyday activities. This will both facilitate comparison between models and allow the features most obviously applicable (or inapplicable) to everyday activities to

be selected as a first step in the development of a model for reasoning about everyday activities.

Within artificial intelligence, two major opposing perspectives for reasoning about action are dominant. *Classical Planning* views activity as entirely explained by the construction and execution of completely detailed program-like plans. In this model, a goal is given to the planner, which then computes a plan that can be carried out in a particular starting situation to achieve that goal. Its antithesis, *Universal Planning*, argues that ongoing activity emerges from a series of spontaneous reactions to recognizable situations. The following Sections describe and analyze these models.

3.1. Classical Planning

> If, to one who governs himself with caution and patience, times and affairs converge in such a way that his administration is successful, his fortune is made; but if times and affairs change, he is ruined if he does not change his course of action...Therefore, the cautious man, when it is time to turn adventurous, does not know how to do it, hence he is ruined.
>
> – Machiavelli, *The Prince*

> The best-laid schemes o' mice an' men
> Gang aft a-gley,
> An' lea'e us naught but grief an' pain,
> For promised joy.
>
> – Robert Burns, *To a Mouse*

Historically, research into the computational implementation of activity has descended from the *Classical Planning* perspective. Classical planning involves the use of a search algorithm to examine a problem and produce a strict sequence of actions (a *plan*) that will solve the problem. The classical planning model is extremely pervasive in AI, and many of the terms used when reasoning about activity are directly associated with it. For example, the use of the term *plan* in an AI paper almost automatically invokes the concept of program-like structures, as opposed to incomplete resource-like structures such as the recipes described in Chapter 2, which would also colloquially be termed plans. Similarly, all types of reasoning about action in AI are still normally grouped under the umbrella term of *planning*, regardless of whether plans in the classical sense are seen to play an important role or not. Referring to *planning* can currently mean anything from reacting to specific situations (which seems to have very little at all to do with even a dictionary definition of planning) to creating, using, and optimizing detailed plans [McDermott and Hendler, 1993] in the classical manner.

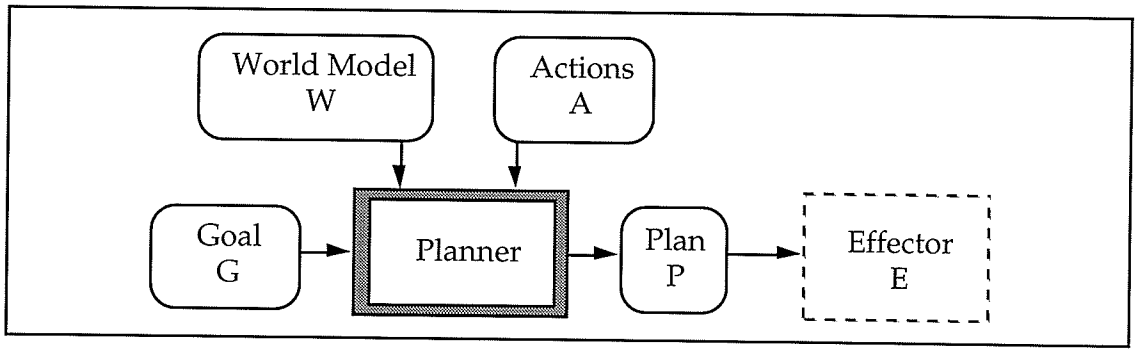

Figure 3-1. General structure of a classical planner (based on [Wilkins, 1988]).

Figure 3-1 illustrates the general structure of a classical planner. The planner is given a collection of possible operations and a problem to solve, and uses a model of how the world around it operates to search the space of possible operator combinations to arrive at a sequence that will cause the goal to be achieved. The plan is then sent to some type of effector to be carried out, much as a computer would execute a program. More formally, a classical planner may be described as a five-tuple $\{U,W,A,P_p,g\}$, where:

- $U=\{s_i \mid s_i = \{d_j\ (j=1..n)\}\}$ is a universe of *states* s_i. Each state is a description of the world around the planner at a particular time, and consists of a set of predicates d_j in some description language. Each state is complete and consistent. This description of U assumes that each predicate or its logical negation appears in each state. The *dimension* of U is n, the number of predicates necessary to completely describe a state. We call a set of predicates $\{d_k\ (k= 1..m, m<n)\}$ a *partial state description*, and denote the set of all possible partial state descriptions by S_p. Note that any $s \in S_p$ describes a subset of U.

- $W= \{s_i \mid s_i \in U, \exists\ s_{init}, s_{curr} \in W; init \leq i \leq curr\}$ is a description of the world around the planner consisting of an *ordered* set of one or more states s_i. One element of W is designated as the *initial state* s_{init}, representing the state of the world before the planner alters it in any way. W also contains a *current state* s_{curr}, indicating the state of the world at the current point in time (the last item of W). Note that initially, $s_{init} = s_{curr}$.

- $A = \{a_i \mid a_i = \{l_i, p_i, e_i\};\ p_i, e_i \in S_p\}$ is a set of *actions*, each of which represents a potential transformation the planner can make to the world around it. Each action is a tuple consisting of a set of *preconditions* p_i and a set of *effects* e_i. p_i and e_i are each subsets of a state. Each action also contains a *label* l_i, that uniquely identifies the action. An action a_k is considered *applicable* when $p_k \subset s_{curr}$ when any variables in the d_i that make up p_k have been appropriately bound to specific constants. We define $A_{curr} = \{a_i \mid p_i \in a_i \subset s_{curr}\}$ as the collection of all such *currently applicable actions*. By definition, $A_{curr} \subset A$. The *application* of an action $a_i \in A_{curr}$ performs a transformation $W \rightarrow W \cup s_{curr+1} \mid s_{curr+1} = (s_{curr} \cup^* (e_i \in a_i))$, creating a new current state by applying the necessary changes detailed in e_k. $\cup^*$ is defined as *union under consistency*, an operation that allows the non-monotonic union of two or more states. $\cup^*$ allows predicates in the latter of the two states on which it operates to replace those in the former (e.g. $p \in s_i, \neg p \in s_j \rightarrow \neg p \in \{s_i \cup^* s_j\}$ (j>i)). In practice, this $\cup^*$ operator is often implemented by representing the effects of an action (e_i) using two separate lists of predicates: one list consists of predicates to be added to the current state, and the other predicates to be deleted (e.g. [Fikes and Nilsson, 1971]).

- $P_p = \{l_k \mid l_k \in a_k \in A;\ k=1..curr\}$ is a *partial plan*, that consists of an ordered sequence of action label components l_k. Each time an action $a_k \in A_{curr}$ is applied, the application process performs a transformation $P_p \rightarrow P_p \cup (l_{curr} \mid l_{curr} \in a_k \in A_{curr})$. Initially, $P_p = \{\varnothing\}$.

- $g = \{d_i \mid i \leq n\}$ is a *goal*, and consists of a conjunction of predicates describing characteristics of desirable state(s) of U. Thus $g \in S_p$, and g defines $G = \{s_i \mid s_i \in U, g \subset s_i\}$, the set of states of U that can be partially described by g, and known as the set of *goal states* of the planning problem. By definition, $G \neq \{\varnothing\}$.

The planner searches the space of U, using the actions A to make transformations to W, until a state s_g is generated. The ordered set of labels stored in P_p is then a complete plan that solves the planning problem. Should $A_{curr} = \{\varnothing\}$ and $s_{curr} \notin G$ any time during this search process, the

planner must *backtrack* by reverse-mapping one or more actions that have been applied to W, in order to try an alternate search path. The planner must also backtrack when a loop or cycle in the search is found (i.e. $s_{curr} = s_i$ (i < curr)). When the complete sequence P is found, it is then sent to some effector E to be carried out in the real world, where, assuming the semantics of the planner are correct, the transformations contained in P will have the same effect on the real world that were performed on W. There is some variation on this formalism between classical planners, but most early planners (e.g. STRIPS [Fikes and Nilsson, 1971]) follow this formalism fairly strictly.

As mentioned above, the classical planning model is a historical one in AI. This formalism is based largely on McCarthy's [1959] *situation calculus*: each state described above is termed a situation, and planning involves logical inference on situations. The most obvious difficulty in this formalism is the amount of search involved in finding P. The general principles of classical planning evolved from early work on theorem-proving (e.g. [Green, 1969]), automated deduction (e.g. [Newell, Shaw, and Simon, 1963]), and other applications of what is commonly known as *problem-solving*[29]. Applying models of problem-solving to activity is difficult, since the the plans produced by theorem-provers are not "carried out" in any sense of the term, but constructed in the background. As such, there is no time pressure on a planning agent: a mathematical problem could validly take a lifetime to solve. The plans produced by theorem-provers are also ends unto themselves: the proof itself is the solution to the basic problem. In any physical world, the plan produced must be carried out or executed in some way if the original goal is to be achieved. The fact that the plan is produced completely ahead of time forces it to be executed in a manner similar to a computer program: each step must be specified completely and all information needed for execution must be included with the plan. The metaphor of plans-as-programs is so pervasive that early classical planners used high-level programming language statements to encode plans (e.g ALGOL [McCarthy and Hayes, 1969]).

The common view of many in the area of classical planning was that formalisms for theorem proving would readily adapt themselves to more general problems [Fikes and Nilsson, 1972]. The first such system, GPS (the

---

29 Nilsson [1971] provides an overview of many behaviours that fall into this category, and common techniques AI uses to replicate them.

General Problem Solver) [Ernst and Newell, 1969], was based on studies of human cognition in solving logic, cryptarithmetic, and chess problems [Newell and Simon, 1972]. While GPS nicely modelled the type of reasoning used in these situations, the search space U in the types of problems studied was always reasonably small. For any significant search space, the amount of time required to find P quickly becomes prohibitive.

Later research involved adopting domain-independent heuristics to improve the general approach, either by making it more efficient or enlarging the class of problems that could be dealt with [Sacerdoti, 1988]. One important development was the realization that actions can be hierarchically abstracted: the planner can begin by applying a general action that in turn can later be broken down into other, simpler actions. This is accomplished by adding to the basic classical model a set of mappings from abstract to atomic actions, and is known as *hierarchical planning.* Hierarchical planning systems such as *NOAH* [Sacerdoti, 1974] can significantly reduce the amount of backtracking and thus the amount of search required by building complete abstract plans, and then using the mappings provided to specify the actions in more detail. A second and more significant development was the realization that P does not have to be constructed in a linear fashion. *Non-linear planning* (e.g. *NONLIN* [Tate, 1975, 1976], *ABTWEAK* [Yang et al., 1991]) allows actions within P to remain unordered until other actions added to P force an ordering. Removing the linearity restriction allows many problems where the conjunctives of g interact to be solved. However, despite these advances, many negative complexity results can still be shown. Chapman [1987] has shown that hierarchical, nonlinear planning is still an NP-hard problem[30], while Bylander [1992] has shown that extremely severe restrictions must be made on both the actions A and the domain in which the planner resides in order to guarantee tractability. Thus, while useful in studying problem-solving in certain specific situations, the practicality of these approaches for complex problems is doubtful.

The difficulties in making practical use of classical planning systems arise largely from differences between most physical domains and the artificial

---

30 This reference also provides an excellent summary of the capabilities and limitations of the classical approach.

theorem-proving domains from which the paradigm was adapted. Some of the more obvious of these are illustrated in Figure 3-2[31].

Many of the practical limitations of classical planners come from the fact that they were designed for static worlds. The real world, on the other hand, is extremely dynamic: changes can occur at any time. This dynamic nature works against any agent that produces detailed plans in the classical manner and then executes them separately. The chance of a spontaneous event occurring that invalidates some or all of a plan increases with the amount of time spent planning, limiting the approach to simple problems.

Theorem-Proving Domains	Physical World
Error-free execution	Errors may be commonplace
All information available at planning time	Must inquire about unavailable information, or perform without it
Closed environment	Others interfere with our work Things don't always go as expected
Search time is not a consideration in planning	Time to plan ahead varies
Perfect model of the world	Incomplete, inaccurate knowledge
All action effects can be detailed	Action's effects depend on situation
Problems are distinct and performed one at a time	Problems are ongoing and many demand attention concurrently

Figure 3-2. Differences between artificial planning environments and the physical world.

Other difficulties arise because agents are necessarily resource-bounded. Because of this, a planner cannot have complete, correct knowledge of the physical world. Plans are thus unreliable, and mechanisms not provided for

---

31 These are differences between the environments for which classical planners were designed and the physical world, and independent of any problems encountered with adapting classical planners to everyday activity. The latter will be discussed in the next Section.

in the classical model must be in place to gather information during planning. Further difficulties in the practical application of classical planning lie beyond the obvious conflicts illustrated in Figure 3-2. For example, many of the actions one can carry out in the real world cannot adequately be described by the state-change mechanism inherent in classical planning [McDermott, 1981].

It is important to note that despite these limitations, the general classical planning formalism is valid and useful in domains that obey these restrictions. Such domains include the chess and cryptarithmetic problems mentioned in Chapter 2. It is also possible to apply classical planning to more complex situations, with the understanding that difficulties will arise if the restrictions in Figure 3-2 cannot be relied upon. Desimone [1992], for example, illustrates the application of a classical planner to military operations planning, but also describes many difficulties in scaling the application due to the restrictions of classical planning[32].

3.1.1. Classical Planning and Everyday Activities

Everyday activities share the characteristics of what Lyons and Hendriks [1992] term a *reactive planning domain*: they require perception to cope with a changing environment, timely decision-making, reasoning about uncertainty, and continual interaction with the environment[33]. The support for each of these in the classical planning model is minimal. From a domain-independent viewpoint, classical planning is useful in situations where behaviour can be completely planned in advance, but is difficult to put into practice because of differences between the artificial worlds for which it was developed and the real world. From the point of view of performing everyday activities, classical planning is a completely unsuitable model. As described in Chapter 2, everyday activities require goal-directive, adaptive behaviour. Classical planning is clearly goal-directed, but is not adaptive enough to be useful in most cases. None of the examples described in

---

32 In the military planning domain, Desimone notes particular difficulty with reasoning about resources, time, and the interactions of multiple agents.

33 Chapter 2 has already shown additional characteristics of everyday activities beyond those that characterize a reactive planning domain. These characteristics, as well as those that arise from the examination of the applicability of existing planning models to everyday activities, are summarized in Section 3.6.

Chapter 2 illustrates the use of detailed plans; rather, I have argued that humans make use of much more general forms of plans in everyday activity. There is a very good reason for this: there is little motivation to construct completely detailed plans when the environment is too dynamic to carry them out [McDermott, 1981]. This is also true for non-everyday types of activities: for example, it has been found that even in structured domains such as solving chess problems, plans that are too detailed may in fact be detrimental to performance [Wilkins, 1980].

Another limitation of the classical model from the point of view of everyday activity is its lack of plan memory. Humans perform everyday activities quickly and with great ease, not because we are especially quick at constructing plans, but because we have performed them many times before. The classical model contains no mechanism for reusing plans (or indeed, making any use of memory except for a model of the world around the planner and the operations it is allowed to perform, which are essentially permanent structures). The extensive reliance on memory required by everyday activities is simply not provided for within the classical planning model, making the use of this model as a basis for everyday activities out of the question[34].

The philosophical basis that underlies the classical planning approach is also fundamentally inconsistent with much of the nature of everyday activities. The main argument for the use of the classical planning model in any given domain is its logical completeness and consistency, making the classical approach a good choice when optimal plans are required. However, as described in Chapter 2, everyday activity does not reflect these ideas. It is neither complete nor consistent, and an agent in such a domain is a satisficing agent, which must do the best it can given the time, knowledge, and other resources available to it. The application of a model that is so

---

34 Simon [1969] provides a wonderful parable which applies here, describing the work of two watchmakers, Tempus and Hora, both of whom are constantly interrupted. Hora constructs his watches in subassemblies, allowing work to be resumed after an interruption. Tempus, on the other hand, constructs his watches from individual parts, and thus the entire watch falls to bits each time he is interrupted. Hora manages to construct many watches, while Tempus accomplishes little useful work, because he is constantly re-doing previous work that was interrupted. The lesson to be learned is that a complex task cannot be handled realistically without some form of memory or state: in this case, the subassemblies act as memory for the parts of the plan that have already been constructed.

philosophically opposite in character to everyday activity seems questionable at best. Minsky [1981] expresses this well:

> I do not believe that consistency is necessary or even desirable in a developing intelligent system. No one is ever completely consistent. What is important is how one handles paradox or conflict, how one learns from mistakes, how one turns aside from suspected inconsistencies.

I do not believe this argument holds universally, but it is certainly both true and obvious for the everyday activities described in Chapter 2.

3.2. Universal Planning

> The general who wins a battle makes many calculations in his temple before the battle is fought. The general who loses a battle makes but few calculations beforehand. Thus do many calculations lead to victory, and few calculations to defeat; how much more no calculation at all!
>
> – Sun Tzu, *The Art of War*

> Fools are fond of hurry: they take no heed of obstacles and act incautiously.
>
> – Baltasar Gracián, *The Art of Worldly Wisdom*

Many of the difficulties with classical planning hinge on its inability to respond to dynamic changes in the environment around the planner. Not surprisingly, then, one of the first variations on classical planning involved anticipating possible changes, and planning ahead for them. This became known as *conditional planning* [Schmidt, 1985]. An example of such a plan is illustrated in Figure 3-3[35].

This Figure shows a complete plan as a series of partial plans plotted over time. Each branch (decision point) in the structure represents some possible difference in the world to be anticipated: this difference could be due to an error (e.g. one partial plan if an action works, another if it doesn't), or due to lack of information (e.g. one partial plan if a diner orders one dish, another for a different dish). Clearly, the smaller each partial plan, the more responsive the agent using the conditional plan will be to the environment. Equally clearly however, the smaller each partial plan, the larger the number

---

35 The structure of conditional plans bears a great deal of similarity to structures for temporal reasoning known as *Chronsets* [McDermott, 1978], which serve essentially the same purpose as conditional plans: conditional reasoning about the future.

of choice points and the larger the number of overall partial plans. It is from this principle that the *universal planning* model arises.

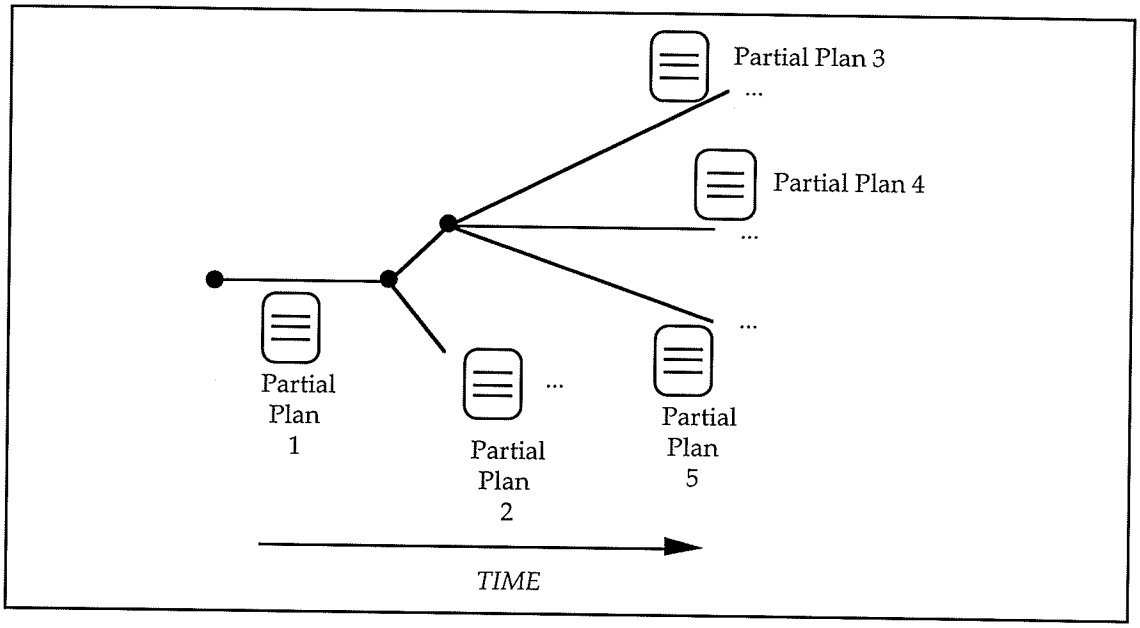

Figure 3-3. Structure of a conditional plan.

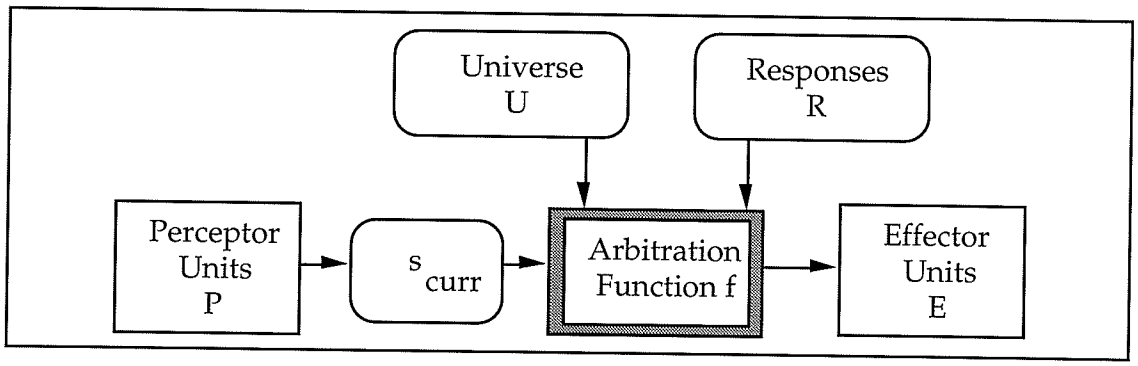

Figure 3-4. Structure of a universal planner.

A universal planner is essentially a conditional planner taken to the extreme: each partial plan becomes a single action, and the choice of an individual action (a reaction to a specific situation) leads to a decision between a number of new actions. Rational sequences of actions thus arise from the goal-directed selection of reactions to possible situations [Schoppers, 1987]. In a typical universal planning agent (Figure 3-4), a set of responses to each possible situation the agent could encounter is pre-compiled. The agent's perceptual

system describes the current state of the world to the agent, which then selects the appropriate response and carries it out. This results in changes to the world, which are in turn perceived by the agent, continuing the cycle. Thus, a universal planning agent relies on knowing the correct response to any given situation, rather than creating a plan and carrying it out.

More formally, a universal planner may be described as a five-tuple $\{U,R,s_{curr}, G, f(x)\}$, where:

- $U=\{s_i \mid s_i = \{d_i\ (i=1..n)\}\}$ is a universe of *states* s_i. Each state is a description of the world around the planner at a particular time, and consists of a set of predicates d_i in some description language. Each state is complete and consistent. The *dimension* of U is n, the number of predicates necessary to completely describe a state. We call a set of predicates $\{d_i\ (i= 1..m, m<n)\}$ a *partial state description*, and denote the set of all possible partial state descriptions by S_p. Note that $s_p \in S_p$ defines a subset of U. This description of U assumes that each predicate or its logical negation appears in each state.

- $R=\{r_i\}$ is a set of responses, each of which is a single command that can be recognized by an effector unit E. Responses differ from actions in that they are directly executable atomic commands, while the response component of an action may consist of any number of changes to the world and (in the case of hierarchical planners) may be highly abstract.

- $s_{curr} = \{d_i \mid i=1..m \leq n\}$ is a set of predicates partially describing the current state of the world around the agent. $s_{curr} \in S_p$ is provided by a set of perceptor units P.

- $g = \{d_i \mid i=1..m \leq n\}$, a *goal* is a set of predicates partially describing a desired state of the world around the agent. This goal may be similar to a goal in a classical planning system (m approaches n), or may consist only of a few factors it is desirable to achieve (m approaches 0).

- $f(x)$: $s_{curr} \rightarrow r \in R$ is an *arbitration function*, resulting in a response the planner can make to the partial state description s_{curr}.

During the operation of a universal planner, the perceptor units provide s_{curr} to the arbitration function, which provides an appropriate response to that state, altering the world the agent inhabits. The perceptor units then

sense a different s_{curr}, and rational activity, it is claimed, results from continuous repetition of this sequence.

Within this general model, a great deal of variation exists between systems, mainly in terms of the implementation of the arbitration function. f(x) has been implemented in various systems as a boolean network [Agre, 1988; Chapman, 1990], situated automata [Kaelbling and Rosenschein, 1991], decision trees [Schoppers, 1987], or networks of actions in a more traditional knowledge representational mechanism [Maes, 1991]. Methods of constructing f(x) also vary: the situated automata of Kaelbling and Rosenschein are meticulously synthesized by hand, while Schoppers' research is mainly in the area of automatic synthesis of universal plans [Schoppers, 1987]. Agre and Chapman, on the other hand, combine both methods through a specification language that automatically generates boolean circuits [Agre, 1988; Chapman, 1990]. Universal planners also differ in the degree to which sensors are assumed to be complete (i.e. $s_{curr} \in U$ as opposed to $s_{curr} \in S_p$, or more specifically, how small the difference n-m is, where $s_{curr} = d_i$, $i=1..m$, $m \leq n$, $n = \dim(U)$).

Universal planning takes a radically different view of activity than that proposed in classical planning, one that concentrates entirely on stepwise reactions and ignores stored plans. In practice there is also some variation on this theme, from theories that include plans in a very restricted form (for example, Suchman's [1987] *situated activity theory*) to systems that use no plans whatsoever (e.g. Agre and Chapman's [1987] *Pengi* system). The universal planning perspective has evolved partly due to the time constraints inherent in classical planning: the assumption is made that the arbitration function f(x) will be faster in selecting an appropriate action than using a planning algorithm to generate a complete plan. In many cases, of course, this will be entirely true; the question is whether reactions alone will suffice to produce the elaborate behaviour of which humans (and even classical planning systems) are capable.

Another major influence on universal planning was the realization that a number of the practical problems associated with classical planning can be attributed to an overly simplistic view of a planner's relationship to its environment. Prior to this, the actions of a planner had traditionally been viewed as a series of alterations to the world around the agent (for example, picking up a block causes that block to be removed from the ground). However, the universal planning approach stresses the connections between the planner's behaviour and the setting in which it occurs (what Agre [1988]

terms the *dynamics* of the situation). As mentioned in Chapter 2, behaviour and setting are intimately entwined with one another: the setting in which behaviour occurs ultimately constrains the behaviour of the planner, while similarly, the behaviour of the planner constrains and alters the settings in which it takes place [Barker et al., 1978; Barker, 1968]. Through stepwise interaction between the planner and its environment, coherent patterns of behaviour emerge, thus displaying *emergent functionality* [Maes, 1991]. Maes [1991] presents empirical data illustrating this property in a simple domain[36].

Despite the fact that universal planning is largely thought to have arisen completely from the inadequacies of the classical model, there is a historical link between classical and universal planners. One of the earliest classical planners, STRIPS [Fikes and Nilsson, 1971] made use of structures known as *triangle tables* [Fikes et al., 1972] to provide for more flexible execution of classical plans. Triangle tables are lower triangular arrays representing the preconditions and effects of each action in a specific plan. Row i in the table corresponds to action a_i in the plan, and the preconditions for a_i provided by the initial state s_{init} form the first cell of the row. The following cells consist of preconditions for a_i that are provided by previous actions $a_1..a_{i-1}$. The effects of a_i are recorded in appropriate cells in later rows in the table. This effectively forms an indexing structure, noting dependencies between actions, that allows the appropriate action to be selected based on the current state of the environment. This is a primitive form of f(x), and the plan itself a primitive form of universal plan, in that if the assumption is made that the preconditions describe a complete state s_i, all possible actions for all possible resulting situations are defined. The main difference between triangle tables and universal plans is mainly one of scope, in that a triangle table is a primitive universal plan for a restricted subset of U; the only states that can be recognized and reacted to are those that form a part of the overall plan. True universal planning makes the stronger claim of representing all possible world states. Universal plans also make reaction to the environment much more explicit than the triangle table paradigm [Schoppers, 1987].

---

36 The reasoning behind the idea of emergent functionality is much older than its application in universal planning. Simon [1969], for example, states that "Man, viewed as a behaving system, is quite simple. The apparent complexity of his behaviour over time is largely a reflection of the complexity of the environment in which he finds himself". Emergent functionality is also an important aspect of other areas of AI, the most obvious being neural networks and other forms of connectionist computation.

Universal planning is still a relatively new field, and there are few good critiques of universal planning available[37]. However, there are many obvious areas to which criticism can be drawn. The first question that needs to be asked of any universal planner concerns the nature of the arbitration function f(x). Ginsberg [1989] constructs an argument against the implementability of f(x) on the basis of the number of universal plans and the size of a boolean circuit needed to construct the function. Greatly predating the development of universal planning, McCarthy and Hayes [1969] dismiss an early attempt at achieving general intelligence through the development of a large and evolving finite automaton through essentially the same argument: they argue that since a typical computer program regarded as a finite automaton might have 2^{10^5} to 2^{10^7} states[38], automata represent a philosophically adequate mechanism for representation of behaviour, but not a practical one. Firby [1992] also points out that storing all possible situations limits the paradigm to simple environments. In defense of the field, there are those who are advocating the use of universal plans as "caches" rather than as a complete architecture with the ability to handle every situation [Schoppers, 1989]. This trend is not yet a common one, however.

At a more basic level, the efficiency of using f(x) as compared to the creation of a complete plan is due to the fact that the function is already in possession of the correct response r to any given s_{curr}: the search required by the classical planner does not have to be performed. This is often touted as the basic advantage of universal planning, but it is largely a misrepresentation: the search is simply performed ahead of time during the compilation of the network, table or other structure that drives f(x). This eliminates much of the time saving: certainly time is saved in that once the appropriate response is computed, it can simply be stored. However, a great deal of time must be spent in enumerating all the individual cases that comprise a universal plan and compiling appropriate responses to all of those cases in order to implement f(x). Using the terminology of [Ginsberg, 1993], this is simply substituting *compile-time* control for the *run-time* control of a classical planner. Assuming that the universal planner is operating in a complex

---

37 A good short article is [Ginsberg, 1989].

38 The criteria for such a computer program is not described by McCarthy and Hayes; nor is their method for estimating these figures. It can be generally agreed, however, that the number of states necessary for an automaton used to represent a program of this size would be vast.

environment, this set of cases will be large, and many of the compiled responses will never be used, simply because the s_{curr} they are associated with are invalid or highly unlikely to occur. In addition to the size of the implementation of f(x), Hayes-Roth [1993] points out that for any significant problem, it may also be computationally unfeasible to obtain enough sensory information to properly gauge the current state in a limited amount of time.

The complexity that lies beneath the surface of universal planning should not be surprising; in the complexity results for classical planning mentioned in the previous Section, Bylander [1992] emphasizes that the complexity lies in finding the sequence of actions to accomplish a goal, rather than in the process by which those actions are generated. That is, whether a system is purely reactive or purely strategic, it is equally difficult to achieve goals optimally through the application of actions. The best that can be hoped for is a heuristic method: one that is tractable, but not necessarily guaranteed optimal.

Another conceptual and practical difficulty associated with the universal planning model is that universal planners maintain little or no model of the world around the agent. The systems of Agre [1988] and Chapman [1990], for example, use a visual search on sensory input from the real world, and maintain no knowledge about sensory information acquired previously from the environment. In fact, the basic universal planning model uses no internal state information at all (resulting in what Genesereth and Nilsson [1987] term *tropistic* agents). This disagrees profoundly with many of the results acquired previously in the study of human problem-solving. For example, there is a large body of evidence indicating that humans make extensive use of cognitive maps (mental models of the world around themselves) when performing problem-solving. Lynch [1960], for example, showed that people could describe the cities in which they live vividly, and make and adapt plans for getting from place to place using only previously acquired, internal knowledge structures. Schwartz [1984] also demonstrates the use of such structures. These mental images are used in many types of everyday problem-solving: we all know, for example, the route from home to work, and what is around the corner when we walk through a darkened but familiar room. The ability to store and process such information is also widely accepted and employed in AI (e.g. [Kuipers, 1978, Yeap, 1988; Hayes et al., 1994]) Moreover, such structures have also been shown to exist in creatures with much less intelligence than humans [Pearce, 1987]; precisely those that the reactive model is supposed to fit so well. These structures are closely related to other forms of mental processes important in problem-

solving, such as imaging and mental projection [Neisser, 1976]. It truly is often more useful to *know* that to *look* [Mason, 1993].

In addition to the conceptual problems brought about by a lack of state memory and model of the world, many practical problems arise due to the necessary overdependency on sensory abilities that goes hand in hand with not maintaining any world model. A high dependence on sensing makes for slow systems confined to simple tasks, and systems that are hypersensitive to environmental changes [Mason, 1993].

Like classical planning, universal planning is highly suitable to a restricted set of tasks. In order for universal planning to perform well, U must be defined completely and a suitable f(x) must be available. Moreover, because of the stepwise nature of universal planning, the domain must be reactive rather than involving long sequences of actions: that is, the domain must be suited to keeping a simple goal in mind and acting on that goal only one step ahead at a time. This is essentially reducing the basic planning problem to the idea of scheduling one action at a time in response to the planner's environment. For forms of activity involving long sequences of actions, the deliberative nature displayed by classical planning makes it in principle the more suitable of the two approaches[39]. Classical planning is slow and cannot recover from errors, but has a more complete picture of activity on which to base its decisions. Universal planners are forced to make a transition from s_{curr} to s_{curr+1} using only g as a guide to the choice of transition. Conversely, classical planners have knowledge of $s_{init} \ldots s_{curr-1}$ as a basis for the choice, and also possibly (assuming nonlinear planning is allowed) some of the states $s_{curr+1} \ldots s_g \in G$. This additional knowledge allows the classical model to make more well-informed decisions.

Also in a manner similar to classical planning, the problems inherent in universal planning models are by no means independent of one another. In many ways, arguing for one model over the other based on a given problem

---

39 In principle, goal-directed reflection should *always* be more suitable than strict reaction in forms of activity involving long sequences of actions. If the domain is particularly dynamic, however, a strict classical model may not be able to keep up with changes invalidating its plans. As a result, a reactive or universal approach would still be more suitable to that particular domain. The success of the universal model in these situations is not due to a general unsuitability of any kind of reflective decision-making, I argue, but to the fact that the classical model requires that a plan be complete before carrying it out. That is, it is a criticism of the classical model rather than of reflective decision-making.

raises the same arguments regardless of the problem chosen. Arguing for or against universal planning, for example, can be a matter of arguing over reaction as opposed to planning. It can also be a matter of arguing over the intense use of world models over the elimination of all forms of state memory. Gat [1993] has argued that these arguments are largely equivalent.

3.2.1. Universal Planning and Everyday Activities

Despite the utility of universal planning to completely reactive architectures, it is no more suitable to everyday activities than is classical planning. While universal planning research argues that coherent sequences of actions emerge from reaction even in complex environments, many refute this view [Cohen et al., 1989; Ginsberg, 1989; Anderson and Evans, 1991]. The most successful examples of universal planning perform in a domain that meets these criteria well: that of video games. In video-game playing, the player generally relies on deciding what to do as situations occur, rather than planning for them ahead of time. Further, most video games are almost purely reactive, and therefore are far beyond the scope of any classically-based approach used to date. The ability to tackle domains such as this has been a large factor in the current trend toward universal planning architectures. Universal planning architectures have also been very useful as an explanation for small behavioural routines that appear in activities such as video-game playing, where individual actions seem internalized as a group, yet appear to have no conscious connection from one action to another while the routine is being performed.

The major barrier to the use of universal planning as a model for performing everyday activities is the fact that the model ignores the basic temporal continuity that forms an important part of everyday activities. Universal planners rely on the assumptions that actions can be performed in isolation - that it simply "makes sense" to perform a given action at a given moment - and that the rationality of any action can be determined using only local information [Hayes-Roth, 1993]. In domains such as video-game playing, this is largely true. However, even those working with this particular domain admit that most activities are not characteristic of video-game playing: they are less hectic, have more complex goal structures, and rely more upon memory and representation than the universal planning architectures provide [Agre and Chapman, 1989]. Even within the domain of video-game playing, it can be demonstrated that the ability to deliberate on decisions when appropriate plays an important role in the ability to perform some video-games well [Kirsch and Maglio, 1992].

Everyday activities do not depend solely on the current stimuli the agent is receiving, as is required by universal planning mechanisms. Rather, they depend on a great deal of knowledge that is largely or completely ignored by universal planners: the internal expectations the agent has for its future, for example, and the actions the agent has completed in the past [Neisser, 1976]. Failure to make use of this global information results in selecting actions that may be locally appropriate, but globally inappropriate [Hayes-Roth, 1993]. Bratman [1987] argues that this coordination between present and future is the most important function of plans, and is the reason why humans must be planning agents. In the performance of everyday activities, global information in the form of internal and external plans is used extensively. The example of cooking described in Section 2.4 illustrates this well. The general internal plan for any cooking task (such as making tea) can be described from start to finish as an integrated entity, as opposed to an isolated collection of reactions. The same is true of every other activity. While reactions may occasionally be involved (e.g. catching a teacup that slipped from one's grasp), they would not be included in any general description of the activity, but instead stay on the periphery to be used when triggered by a certain set of circumstances. While crucial to the correct performance of everyday activity, such reactions are not responsible for the bulk of our behaviour in such activities[40].

Ignoring these resources essentially makes universal planning a direct inverse of classical planning from the point of view of its application to everyday activity: it is certainly adaptive, but is not goal-directed enough to be of use in most cases. One cannot claim that the universal planning model is not goal-directed: it is certainly goal-directed in that it "continually reevaluates what to do" [Agre, 1988] and selects its actions on the basis of its goals. However, this is an extremely naive sense of the term. It agrees with Newell's [1988] principle of rationality[41], but fails to consider the fact that goal-directed behaviour is usually more than just considering what is logical in the extreme short term. As argued in Chapter 2, rationality (and goal-directed

---

40 As in the previous Section on classical planning, I am not arguing that *all* activity takes this form, and nothing is strictly reaction-based. Rather, I am arguing that the bulk of everyday activities as they have been defined in Chapter 2 do not fit the strict reaction mechanism.

41 *Actions are selected to attain the agent's goals.* See Section 2.3.

behaviour) is a matter of both local and global coherency, the latter being a factor impossible to consider in the universal model.

Even in domains where universal planning would seem well-suited, such as a fast-paced activity like tennis, these characteristics appear. The tennis player always has an idea of where the ball is to be hit, for example, and bases that choice on an analysis of his or her opponent, on where previous shots have been hit, and on the success or failure of those shots. Bartlett [1967] explains this concept well[42]:

> Suppose I am making a stroke in a quick game, such as tennis or cricket. How I make the stroke depends on the relating of certain new experiences, most of them visual, to other immediately preceding visual experiences, and to my posture, or balance of postures, at the moment. The latter, the balance of postures, is the result of a whole series of earlier movements, in which the last movement before the stroke is played has a predominant function. When I make the stroke I do not, as a matter of fact, produce something completely new, and I never merely repeat something old. The stroke is literally manufactured out of the living visual and postural schemata of the moment and their interrelations.

In summary, the limitations of universal planning systems from the point of view of everyday activity are mainly due to the fact that they limit their reasoning to the present moment, and to what they can currently perceive. Universal planners can group actions into routines that can be carried out in a given situation, but there is no explicit reasoning from action to action or any explicit consideration of future intentions. If a planner uses no plans, and simply reacts repeatedly to the current situation, then very little can be done in the long term. We can solve many problems by keeping a goal in mind and thinking only one step ahead at a time, but a great many problems we encounter in everyday activity will not adapt themselves nicely to this technique. Maes [1991] and others have performed some work on integrating sequences of actions by effectively including R as a part of s_{curr}. Thus f(x) can then be modified to be able to access previous responses as well as responses that have been historically useful following s_{curr}. Thus far, however, this has been demonstrated to function only in very simple environments, where the agent has only a few possible actions at its disposal and more importantly, where the connections between actions are extremely limited.

As is the case with classical planning, there are also fundamental philosophical differences between the model itself and the nature of everyday

---

42 This is an often-quoted passage, and also appears in [Neisser, 1976].

activities. Much of this is inherent in the argument as to whether coherent sequences of activities can emerge from individual reactions. However, there is another trend in universal planning research that disagrees strongly with the nature of everyday activity. In general, most universal planning work addresses low-level details of activity to which the paradigm is well-suited (e.g. limb motion or navigation), and avoids more complex commonsense reasoning issues [Georgeff and Lansky, 1987]. The universal planning model makes the assumption that the same essentially unconscious mechanisms responsible for these low-level reactions are responsible for much higher-level activity. I argue that there is little basis for this broad assumption, and that in fact, the very nature of everyday activity described in Chapter 2 contradicts it. The low-level components of activity (e.g. muscle coordination) can be explained by the use of motor programs (Section 2.2) controlled by conscious thought processes, and evidence of such conscious, non-reactive thought processes can be found in almost any example of everyday activity.

3.3. Variations on Classical and Universal Approaches

Recently, advocates of the classical approach have concentrated on developing techniques that attempt to extend the basic classical planning formalism. Many of these focus on specific problems with the classical architecture, such as resource management [Tate and Whitier, 1984]. However, the bulk of recent work in classical planning concentrates on repairing and extending plans at execution time. A common method is the interleaving of planning and execution: some form of interrupt is provided, allowing the planner to pause while some portion of the plan it is producing is executed [Wilkins, 1988; Wilensky, 1983; Ambros-Ingerson, 1987]. These interrupts can be implemented in many ways: viewing missing information as "faults" in a plan, for example, can allow planning to be interrupted to carry out a sensory action to fill in the required information [Ambros-Ingerson, 1987]. Other systems allow changing goals and beliefs about the world to alter the order in which plans are executed [Georgeff and Lansky, 1987].

One basic problem with interleaving systems is that the planner must have some intelligent method of deciding when to plan and when to execute. There may be times where planning can go on for a considerable length of time, and others where only a single action can be planned at a time. In this sense, universal planning is a subset of the interleaving model, since reacting to each s_{curr} is the same as forcing the interleaving model to interrupt

planning every time a suitable A_k is found. While present research is concentrating on strategies for control transfer between planning and execution (e.g. [Krebsbach et al., 1992]), the criteria that are used to decide when to plan and when to execute are usually shallow and obvious and make little use of the reasoning behind the plan. Most interleaving systems monitor only the preconditions of the various actions in the plan, and thus can recognize problems only when they have actually occurred. For example, when driving a car, such a system could only stop its course and decide to change direction only after it had actually hit something [Agre and Chapman, 1989].

More advanced forms of these systems have begun to include appropriate sensory processes and other peripheral systems not generally included in classical planners, and allow plans to change somewhat on the basis of the environment. *Phoenix* [Cohen et al., 1989], for example, allows skeletal plans to be fleshed out during execution. It consists of a *cognitive component* that fetches and expands plans, and a *reactive component* that attempts to fit any of a number of reactive rules to the given state of the environment. Should the environment fit one of these rules, the action is carried out, and the cognitive component is informed (via a flag) that the environment has been altered. At a later point, the cognitive component must clear the flag and make the plan reflect the changed environment. Phoenix suffers from two major flaws, however. First, the expansion process is primitive, in that it involves substituting a predefined sequence of actions for the abstract plan, much like the hierarchical planning process described earlier. More significantly, the artificial division of reactive and cognitive components is like the agent's right hand not knowing what its left does. The cognitive component of the agent only finds out that the agent has performed some action long after it has done so. This reaction is based on extreme involuntary responses, such as yanking one's hand back from a burning candle. As I have argued previously, most action in the world is not of this type.

Despite such advances, interleaved planners all make use of classical planning techniques, albeit in a more flexible way. This being so, they still suffer from the problems of intractable reasoning that lie behind classical planning: as Agre and Chapman [1989] point out, if plan construction is a computationally intractable process at the beginning of a task, it is also going to be an intractable process when trouble arises and a new plan is required. Basing all activity upon the construction and execution of detailed plans is

not the viable solution, as the examination of classical planning earlier in this Chapter has shown.

There are also a number of perspectives that are similar in spirit to universal planning. The most well-known of these is Brooks' [1986, 1991] *subsumption architecture*, which can generally be described as an architecture for behavioural control [Firby, 1992]. In the subsumption architecture, various behaviours (e.g. balance on one foot; travel away from light) are implemented independently as finite automata. Those behaviours interact with one another in complex ways, producing emergent functionality. These are similar to universal planners in that all possible behaviours must be programmed and thus suffer similar difficulties. They are also similar in their use of network structures to implement arbitration between alternatives. Universal planning systems use horizontally-organized networks, where s_{curr} enters once side and a response exits the other. The subsumption architecture, on the other hand, uses a vertically organized network, where behaviours form layers, each more complex level subsuming those below it. While complex motor activity has been implemented using this architecture, it is limited by the complexity of the interactions of behaviours: Brooks himself admits that managing more than a few (a dozen) behaviours productively is extremely difficult [Brooks, 1991]. Recent work is concentrating on controlling the size and complexity of such systems. Firby [1992] presents a system that essentially allows a reprogrammable behaviour-based control system, allowing the same behaviours to be used in many different ways without having to encode them completely in advance. The utility of such systems, however, is again an open question.

The approaches described in this Section are by no means the only in existence that make use of aspects of planning and reacting. Recent years have seen a great many developments in this area: a glance at any recent review of modern planning research (e.g. [Hendler, 1992; Lyons and Hendriks, 1992]) reveals dozens of efforts, each slightly different than the others. There is little point, however, in describing all of these here: many have little bearing on what we have defined as everyday activity, and many others are only minor variations on the basic themes described above. The systems described in this Section are, I believe, fair representatives of current trends in AI planning research. Those interested in further details of these approaches, or the many variations present in implemented systems, are directed to the aforementioned reviews.

3.4. Memory-Based Approaches

Although the above approaches all differ fairly radically from one another, and are unsuitable for everyday activity for various reasons, there is one common reason they all share: none captures the experience-based reasoning that Chapter 2 has argued is one of the the primary characteristics of everyday activities. The AI paradigm most commonly associated with such reasoning mechanisms is known as *Case-Based Reasoning*. Case-based reasoning involves recalling and adapting previous experiences and making use of them in new situations [Hammond, 1989a; Reisbeck and Schank, 1989]. To do this, planners employing a case-based approach make extensive use of *episodic memory* to store and refer to the previous plans and the successes and failures that have occurred when they were used. Episodic memory is a type of memory that stores specific episodes of experience or behaviour [Hammond, 1989a; Reisbeck and Schank, 1989; Estes, 1982]. For example, recalling a ticking clock, a birthday party one had been to, or any other memory specific to a particular time and place [Estes, 1982]. Case-based planners recall these specific instances, then alter them to fit a new situation.

Case-based reasoning applies knowledge about specific experiences rather than general task knowledge, simply because generalization loses a great deal of information available in specific experiences [Reisbeck and Schank, 1989]. The approach also argues that abstractions are difficult to remember, while specific episodes are easily remembered, and that general or abstracted knowledge of particular activities (termed *paradigmatic* cases) is of little use to an intelligent agent [Reisbeck and Schank, 1989]. This type of reasoning has proven useful in many situations: law, for example, involves mainly the analysis and application of previous cases; business also shows much of the same types of reasoning [Reisbeck and Schank, 1989].

One particularly relevant effort in case-based planning is the Chef planner, mentioned previously in Chapter 2. Chef makes use of an episodic memory to store its previous efforts (tools and techniques) at cooking, and adapts old recipes to create new dishes. Chef also indexes the failures in its cooking efforts, thus learning from experience and not repeating the same mistakes twice [Hammond, 1989a]. It is an interesting and useful system, but as discussed in Chapter 2, represents the expert behaviour of a chef rather than the everyday activities of an average cook.

The inapplicability of Chef to everyday cooking illustrates much about the general inapplicability of the case-based planning paradigm to everyday activities. Much of the problem lies in the the reliance of Chef (and the

paradigm in general) on episodic memory. There are essentially two components to human memory, episodic and semantic [Estes, 1982]: episodic memory stores knowledge related to specific experiences as described above, while semantic memory stores general facts and descriptions. While episodic memory is indeed important, semantic memory has a much greater role to play in everyday activity than the case-based paradigm allows. An everyday task is performed many times, far more than possible for reliance on episodic memory to be useful. When I describe how I make tea (recall Figure 2-3), I do not remember any one *particular* episode of tea making; nor do I try to sort through them all to find commonalities. Rather, I use an amalgam or generalization of *all* the times I have made tea, resulting in the description of Figure 2-3. That is, everyday activities are based on precisely the paradigmatic cases that case-based reasoning ignores.

Norman [1988] describes semantic memory as being akin to taking multiple exposures of individual people with a camera: after a number of exposures, one has a general picture of the features of the group. The unusual aspects stick out from the photograph (e.g. someone who is especially tall), but the general structure is the most visible and important aspect. As in this analogy, specific episodes of everyday activities are still important on some occasions, but only in the context of the paradigmatic case: for example, one common reminder associated with my internal tea recipe might be to ensure that there is enough water in the kettle before turning it on, because I once ruined a kettle by heating it when it was empty. This is a reminder provided by a specific experience, but this reminder only occurs in the context of the general internal tea-making recipe. When performing everyday activities, specific episodes that come to mind are generally those of especially strong intensity or significance, such as the example above: like an especially tall person in the camera analogy above, they "stick out". This does not imply a significant reliance on episodic memory, however. The general case still forms the main body of our reasoning in everyday activity, and be viewed as as an amalgamation of episodes of behaviour, just as the resulting multiple exposure describes an amalgamation of all the portraits. Schank and Abelson [1977] term such general structures *scripts*[43].

---

[43] Scripts also contain important links to other information necessary to participate in an activity, just as recipes evoke reminders of specific pieces of knowledge required for their application. Scripts also, however, contain aspects which are unsuitable for everyday activity. Chapter 4 describes scripts and their relationship to everyday activities in greater detail.

The emphasis on semantic over episodic memory is not supported or advocated by the case-based reasoning model, and thus while the spirit of reasoning from experience is critical to everyday activity, the model itself is largely inapplicable. Since it is clear that generalized structures are important to everyday reasoning, the emphasis on episodic structures in case-based reasoning seems inappropriate. The reason for this is that case-based reasoning is concerned as much with learning (integrating new experiences into memory and applying those to unfamiliar situations) as with making use of these memory structures in a general way. As can be seen in any theory of memory (e.g. [Schank, 1982]), building a general or paradigmatic case from many individual episodes is a complex task - harder, many would argue than simply storing cases and attempting to modify the most appropriate to a new situation.

3.5. The Spectrum of Activity

Each of the above models is suited to a particular type of activity: classical planning to static domains and well-defined, distinct problems; universal planning to reactive types of activity where highly-compiled domain-specific knowledge is readily available; and case-based planning to those activities that rely extensively on adapting specific instances of previous behaviour to novel situations. Equally obviously, all of these models begin to break down outside of domains in which their particular restrictions hold. In particular, I have argued that everyday activities are beyond the scope of all of these.

The reason for the limitations of the above models is that each was developed from the examination of very specific forms of human activity. In general, the various forms of activity in which humans regularly engage form a wide-ranging spectrum, as illustrated in Figure 3-5. The existence of such a spectrum is only now beginning to be acknowledged by those in AI (e.g. [Hayes-Roth, 1993]). This spectrum varies along many dimensions: activity may rely heavily on plans, or use no plans at all; a planner may reason explicitly about action when deciding what to do, or that reasoning may be hidden in some action arbitration function, as it is in universal planning. The domain may require quick responses to events, or there may be time for considerable reflection on the part of the planner. The domain may also require little in the way of perception or feedback, or may be perception-intensive; and activity may be almost entirely routine, or novel situations may be the norm.

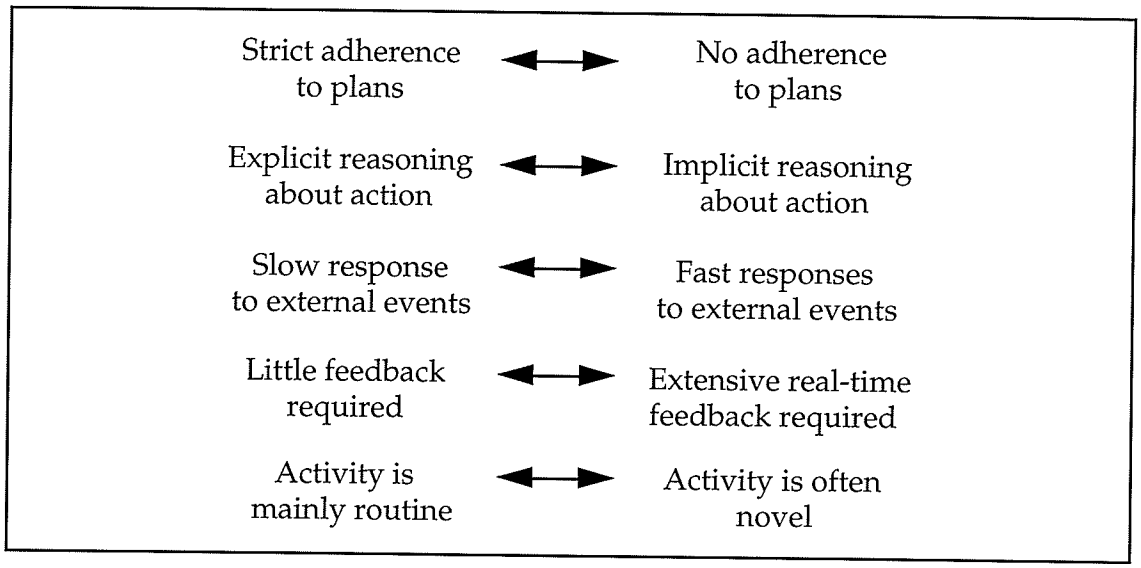

Figure 3-5. Some of the dimensions of activity.

Concentration on one particular part of the spectrum will result in a very different model of activity than that produced by concentrating on a different part of the spectrum. Consider, for example, the use of plans. One might speak of a plan to acquire money (by, say, getting a job or robbing a bank). One might speak of creating or using a route plan for driving from one place to another, or a plan to get a series of factory orders completed in optimal time. One might also speak of a written set of instructions for building a house (a blueprint) or for preparing a meal (a recipe). These are all very different types of plans, although they do have some features in common: they are means to an end, and will consist of a series of knowledge that the planner can use as a resource to bring it closer to that end. Different kinds of plans are also suitable in different situations: a blueprint would not be necessary for a plan to cross the floor from one side a room to the other, while an informal verbal plan would be unsuitable for a plan of attack during an armed conflict.

Cognitive models for making use of each of these types of plans differ widely, simply due to the widely different types knowledge these plans contain (and the amount of knowledge they omit). Some plans can be followed closely, while others must be diverged from frequently and require varying amounts of "generally understood" knowledge to be applied to a given activity. As discussed in Chapter 2 for example, a recipe may have a step such as "add eggs", neglecting to mention that the shells are not to be included for the simple reason that this will virtually always be the case. In general, the more

incompletely specified or abstract the plan, the greater the additional knowledge required by the agent to make use of it[44].

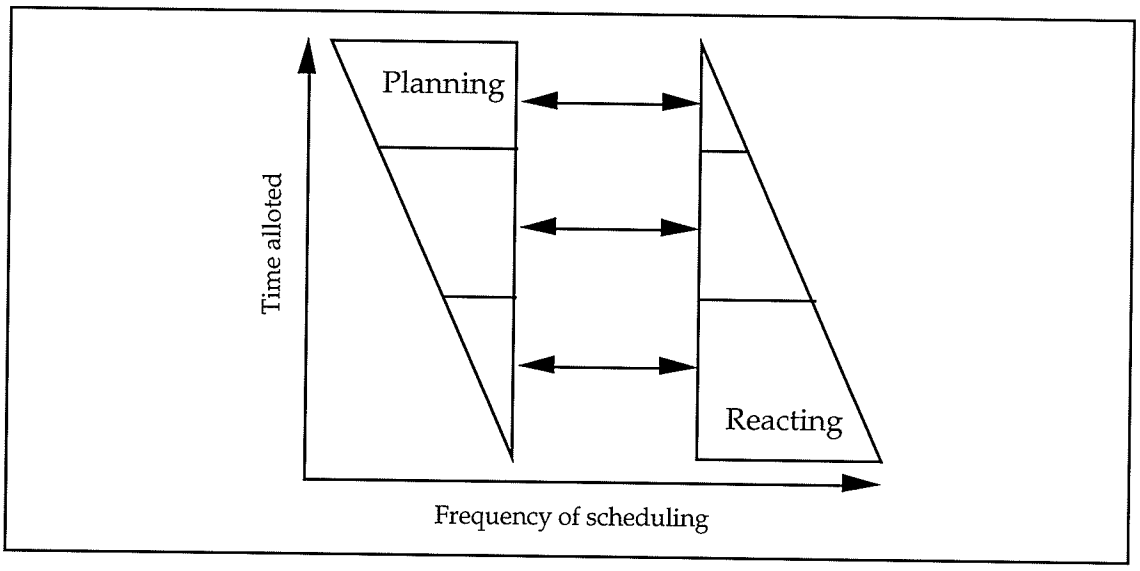

Figure 3-6. Relationship of timely response to dependence on plans [Lyons and Hendriks, 1992].

The dimensions of the spectrum of activity shown in Figure 3-5 are also by no means independent of one another. The amount one can rely on planning, for example, depends directly on how critical timely responses are in the type of activity being performed. Hendler (in [Lyons and Hendriks, 1992]) was one of the first to explicitly recognize this relationship, shown in Figure 3-6. Hendler makes use of this spectrum in his *APE* architecture, which supports five levels of activity, each of which has more planning that the one before it (showing aspects of Brooks' [1986] subsumption architecture and Cohen et al.'s [1989] Phoenix system, both previously described). Problems that are not solvable by lower levels are passed to higher levels, supporting timely responses as well as planning when required [Lyons and Hendriks, 1992]. The difficulty with this approach is that it recognizes the idea of a spectrum in spirit, but supports only discrete levels of planning and reacting: a given

---

44 Despite the fact that most early planning research was performed from a domain-independent perspective (e.g. [Tate, 1976; Sacerdoti, 1974]), it is becoming more generally accepted that that the gap between the theoretical difficulty of planning and its practical application is due to the lack of extensive domain knowledge [Bylander, 1990].

problem might not be suitable to any particular combination implemented in APE. In addition to this problem, this architecture suffers the same problem as interleaving systems: to the extent that classical planning is used, all of the difficulties of classical planning are still present. In order to truly support both the adaptive and goal-directed behaviours I argue are crucial to performing everyday activities, as well as the everyday rationality described in Chapter 2, both the use of experience-based plans and the ability to react outside the guidelines formed by those plans must be freely integrated. This integration is one of the most difficult problems still facing planning systems [Lyons and Hendriks, 1992].

The fact that each of the models discussed in this Chapter differs widely from the others is not a problem in and of itself. As Chapter 1 described, many specific theories all have their roles to play in the development of an overall general theory of activity. The difficulty arises when one attempts to apply or extend a specific model to an area of the spectrum to which it does not naturally apply. From the preceding Chapter, it should be clear that the areas of the overall spectrum of activity to which classical and universal planning belong are very different from those occupied by everyday forms of activity. It would thus seem an unusual approach to adapt solutions from these areas to deal with everyday activities. However, this Chapter has demonstrated that is largely the approach AI has been taking.

Even if it were accepted that an approach suitable to a completely different type of behaviour could be adaptable to everyday forms of activity, it would again seem very unusual to choose the classical or universal models as a basis for that adaptation. Chapter 2 described the unusual and rare nature of the forms of activity on which classical planning was based (tasks at the limits of human ability), and the strictly reactive domains in which universal planning performs are almost equally rare. In comparison, everyday activity encompasses much of what we do by its very definition.

3.5.1. General Models of Activity

Rather than modifying classical or universal planning to suit a portion of the spectrum of activity to which they are inapplicable, an alternative would be to develop a more general theory of all forms of activity, thus subsuming the realms of classical and universal planning as well as the everyday activities in which I am interested.

One such theory is provided by Norman [1988]. Norman contends that a seven stage model (Figure 3-7) can be used to explain all of human action. In this model is the goal: the basic notion of what is desired. Having recognized a goal, an agent adopts an intention to perform actions to achieve the goal[45]. A sequence of actions is then developed to satisfy the intention, and they are executed. Following these *execution* stages are three *evaluation* stages. These involve perceiving the world, interpreting those perceptions, and evaluating them in light of what was expected from previous actions [Norman, 1988].

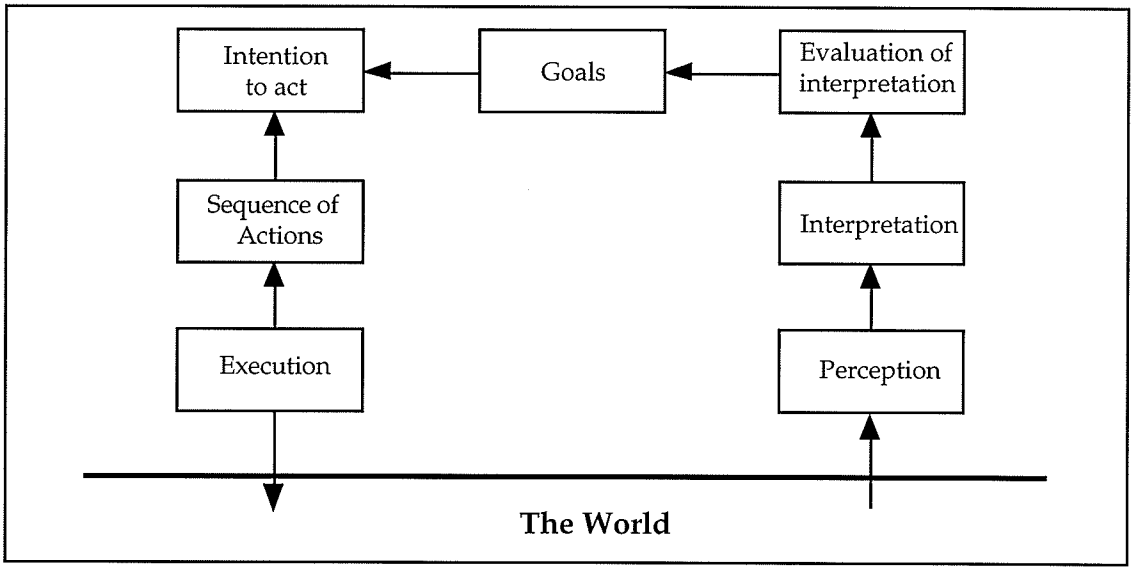

Figure 3-7. Seven stages of activity [Norman, 1988].

This model captures much about activity from a broad perspective. However, in order to make use of the model in explaining any particular activity, one has to violate many of its principles: we sometimes act without looking, or react in a universal planning-like manner without creating sequences of actions. Similarly, the model fails to capture much about everyday activities that is explained in Chapter 2, especially the fact that ongoing activities blend together rather than being treated as distinct entities. In fact, Norman explains that in many examples these stages are not necessarily sequential. This violation does not make the model any less viable for the purpose of human-centred design. However, from the point of view of applying this model to

---

45 To Norman, an intention is any action which is not directly executable.

the human performance of everyday activities, it eliminates much of the purpose of organizing activity into sequential stages.

The difficulty with theories such as this one is that while they capture much of the general principles of activity (e.g. perception, possibly thinking ahead), they say very little about how we go about performing any specific activity. They are like the proverbial knowledge of everyday activities described in Chapter 2: they have a lot to say in general, but are hard to apply coherently to any particular situation. In the case of the above theory, Norman himself admits that it is intended to be an approximate model, useful in explaining human interactions with everyday things, and not a complete psychological theory.

The problem with developing a general approach to all of activity is that we don't currently understand many forms of activity well enough to know the types of mechanisms needed to accomplish them. While we understand certain specific types of activity (e.g. classical planning domains) quite well, others are still largely unexamined. As mentioned in Chapter 2, even everyday activities have been largely ignored by AI until very recently. Developing such a theory is also generally recognized as a big job both experimentally and computationally. Developing a complete system to accomplish even *one* type of activity well can take a great deal of time, and (as we have seen throughout this Chapter) such efforts often say little about other types of activity.

While a complete theory of all forms of activity is one of the ultimate goals in AI, it would seem that it is still a long way off. Thus, at least at the present time, it does not seem to be an appropriate alternative to explain everyday activities through the development of an overall theory. However, I argue that the development of specialized theories of everyday activities will greatly aid the development of such an overall theory, because everyday activities encompass a much more significant portion of the overall spectrum of activity than the activities on which classical and universal planning are based.

3.6. Toward a Model for Reasoning About Everyday Activities

> The good of activity is in its timeliness
>
> – Lao Tzu, *Tao Te Ching*

To this point, I have characterized everyday activities as requiring both goal-directed and adaptive behaviour. The examples in Chapter 2 have shown

that there is a great deal of structure in everyday activities on which goal-directed behaviour can rely: we often perform a routine activity in a particular manner, for example, and the context provided by other activities and the setting in which an agent finds itself imposes restrictions on current activities. Indeed, this very structure is what differentiates everyday activities from other types of activities. Equally evident, however, is the amount of variability in any everyday activities: no matter how many times one makes tea, for example, no two will contain the exact sequence of actions. There may be a routine that is commonly employed, but the dynamics of the setting demand that routine be altered continually as the activity unfolds.

It is the bridge between these two phenomena, the goal-directed and adaptive nature of everyday activities, that has proven to be the major difficulty with attempting to apply current AI planning models to everyday activities. As this Chapter has described, existing architectures are all based on reasoning mechanisms that are either too overconstrained (i.e. requiring a completely detailed plan and lacking the ability to take into account the variability of everyday activities; or making the assumption that every possible reaction can be compiled into a single universal plan) to handle the complex interaction between the goal-directed and adaptive phenomena of everyday activities.

The reason for this, I have argued, is that such approaches are failing to take into account the nature of everyday activities - instead, they attempt to extend mechanisms that work well with particular aspects of non-everyday activities into a realm for which they are inappropriate. On the other hand, I have also shown that the development of an overall grand theory that will explain the complete spectrum of activity is also unrealistic. This does not by any means paint a bleak picture of everyday activity from the point of view of AI. It means only that any theory that is expected to explain how intelligent agents perform everyday activities must begin from the specific perspective of those types of activities. That is, it must begin with an understanding of the characteristics laid out in Chapter 2.

Emphasizing a focus on the specific characteristics of everyday activities does not imply that existing AI planning models are unsuitable for their own specialized environments. More importantly, it does not imply that there is no relationship between everyday activities and other types of activities, or that existing theories have nothing whatsoever to contribute to a model explaining everyday activities. The relationship between existing AI planning perspectives and everyday activities can be seen when we examine in detail the spectrum of activity from the perspective of the novelty of the

activity in question to an agent (Figure 3-8). When performing activities that are completely automated (e.g. automatically releasing a hot cup as it burns one's hand), spontaneous reaction is sufficient to accomplish the activity. This is closely related to performing a completely routine sequence of actions, such as assembling the same sequence of parts repeatedly in a factory. In this case, the sequence of actions that compose the activity becomes so routine, and the knowledge involved so extensively compiled, the number of non-routine situations that can arise is minimal or non-existent, and a universal plan results. The intellectual work involved in carrying out the activity is minimal in this situation, and it would be wasteful to make use of any structure or mechanism that expended more than minimal intellectual activity.

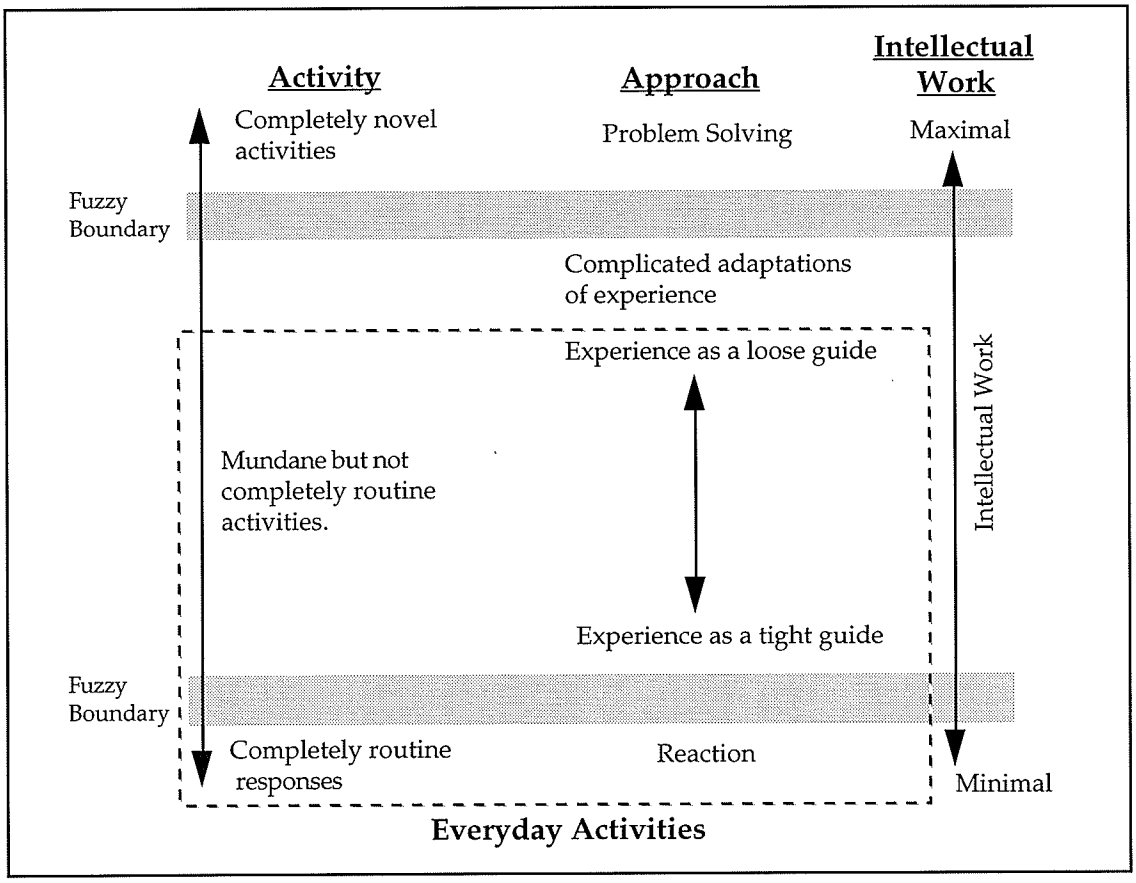

Figure 3-8. Detailed spectrum of activity as it relates to novelty.

At the opposite end of the spectrum, we have activities with which the agent has no previous direct experience: these activities will have little in the way

of compiled, directly applicable knowledge on which to base activity, and thus plans must be developed using problem-solving techniques well-known to AI. In between these two areas, however, there is a wide expanse of activity with which the agent has various degrees of experience. There is no hard and fast boundary where the use of experience turns to problem-solving, or to strict reaction. This is indicated through "fuzzy boundaries" between these levels.

At some point within this expanse, the agent's experience becomes enough that activities have the narrow and/or shallow decision structures that characterize everyday activities (the dashed box in Figure 3-8). Within this boundary, activity takes on the characteristics described in Chapter 2. The agent's experiences serve to guide its activity to varying degrees: at the lower end of the spectrum, to almost universal-plan-like levels; on the upper end, as vague guides for exploring alternatives during the course of activity. In either case, and the range in-between, the knowledge capturing the agent's experience must be applied in a flexible manner: not necessarily dictating what the agent must do (even at lower levels), but guiding the agent through a course of activity, taking into account changes in the environment, the agent's other needs, and future activities the agent intends to perform. Note that the grouping of everyday activities completely envelops the lowest level of spontaneous reactions. This is because while most of what is involved in everyday activities is above this level, there are still aspects that can only be handled by reaction (e.g. dropping a hot cup, as discussed above)[46].

In order for an intelligent system to accomplish this range of activity, its cognitive mechanisms must depart radically from those used in current AI planning models: it must be able to apply knowledge of an everyday activity to the degree that the situation warrants. This means making use of the extensive previous experience the agent has had with the activity when circumstances match what the agent is used to experiencing: that is, applying the most routine methods possible. It also means going beyond those routine methods and using deeper knowledge about the activity when the situation is not completely routine: *the less routine, the less direct experience the agent will have available to it, and the more active search will have to be*

---

46 Similarly, we have already seen that even in in all but the the most abstract and removed problem-solving situations, we must be able to reach across the fuzzy boundary dividing everyday activities from problem solving to be able to adapt the solutions to the changing world.

substituted[47]. Most importantly, it requires an understanding of when such deep reasoning methodologies are useful, and when they will be a waste of effort; that is, when the structure of the activity is strong enough to rely on normative processing methods (i.e. direct experience, or more plan-like structures) and avoid search, and when experience must be used as a more general guide to activity. This is clearly a tall order, but it is also a process that humans make use of almost constantly in their everyday experience; one that meets all the criteria discussed in the previous Chapter for carrying out everyday activities (Figure 3-8). It is commonly known as ***improvisation***.

3.7. Summary

This Chapter has described and analyzed several of the planning models most prevalent in AI from the point of view of their application to everyday activities: classical and universal planning, derivatives of these approaches, and memory-based planning. This Chapter has also presented a view of human activity as a spectrum, and has attempted to place the behaviours to which each particular planning model is well-suited on this spectrum.

Each of these models was designed for a particular type of activity outside of the that portion of the spectrum occupied by everyday activities, and each suits the behaviour it was intended to model very well. However, none of these systems directly attempts to account for the phenomena of everyday activities as described in Chapter 2. Instead, they unsuccessfully attempt to extend mechanisms that work well for one specific type of activity to a completely different area to which they are not naturally suited.

In placing everyday activities along a particular portion of the spectrum of human activity, I have argued that an intelligent system requires both routine knowledge as well as deep knowledge behind its routines in order to accomplish everyday activities. I have argued that humans integrate both of these types of knowledge in performing everyday activities, in an approach commonly known as improvisation. The following Chapter describes

---

47 The amount of search required will gradually increase as direct experience becomes less applicable (that is, as the situation varies more from that with which the agent is familiar). This will eventually lead to the type of extensive adaptations of experience that characterize the case-based reasoning work of Hammond [e.g., 1988], and further down the line, to more direct forms of problem-solving. Long before this, however, these will cease to be everyday activities, using the definition adopted in Chapter 2.

improvisation in detail, illustrating how improvisation can accomplish everyday activities and indicating the cognitive mechanisms improvisation requires of an intelligent system.

CHAPTER 4

IMPROVISATION

We have little to live and much to know, and you cannot live if you do not know.

– Baltasar Gracián, *The Art of Worldly Wisdom*

Daily life is formally an intractable problem: being intelligent isn't predicated on guaranteeing you can stay out of the complexity; it's a matter of functioning well in the midst of it, in spite of it.

– Randall Davis, *A Tale of Two Knowledge Servers*

4.0. Introduction

Just as humans are satisficing agents, they are improvising agents. While we can easily describe how we perform any everyday activity, we never in fact perform it exactly the same way twice. We may have a general methodology that we follow to accomplish a course of activity, but we often stray from this methodology as much as we follow it. We *improvise* on basic methods of accomplishing an activity, using knowledge of that activity and the world we inhabit. This approach gives us the flexibility to adapt to variations in the world around us, and to accomplish our activities despite unexpected obstacles that may arise. Indeed, it is one of the oldest methods of accomplishing activities [Jencks and Silver, 1972].

There are certain activities where this improvisation process is more conscious and obvious than others, and thus those with which the concept of improvisation is more traditionally associated. Jokes and storytelling, for example, are often associated with this type of behaviour, as is theatrical

improvisation [Hodgson and Richards, 1974]. However, the range of activities to which improvisation applies is much wider than this. Jencks and Silver [1972] have made extensive studies of improvisation and *ad hoc* activities. They show that improvised behaviour occurs in a huge variety of activities: music, photography, politics, measurement, cooking, architecture and games, to name a few. They also give specific examples of improvisation in more common activities. For example, assume that I have a pencil that I wish to sharpen, and I have no pencil sharpener. Jencks and Silver [1972] list eleven different methods that could be used to accomplish this activity if one were in a typical kitchen, such as using the edge of a pair of scissors to scrape the wood from the lead; peeling the pencil with a vegetable parer; grating it with a cheese grater; or burning the wood away from the lead.

As can be seen from this example, there is an extremely large range of approaches that one can use to accomplish the same activity. Most of the time, we improvise on our general or normal methods of accomplishing an activity to a fairly slight degree, and it takes very little intellectual work to do so (e.g. substituting scissors for a knife is an obvious one and takes little thought to recall). However, in more extreme situations, we may invest more time delving into the principles behind the activity to perform the activity in more novel ways. In this case, for example, the structure of wood suggests several alternatives (scraping, burning), and the locale suggests possible methods of accomplishing these. The more unusual the alternative, the more intellectual work it takes to think of.

Substitutions such as that above are the most obvious examples of improvisation. This is similar in virtually all everyday activities. In cooking, for example, the thought of improvising immediately reminds one of substituting missing or unobtainable ingredients. In everyday activities, however, there is much more to improvisation than such large-scale substitutions. No matter how often one has performed some activity, there are always small variations: when I make tea, for example, where to place the cup when pouring water into it is not a fixed part of my routine; it differs each time I perform the activity. While in general everyday activities are carried out in a routine, predictable manner, there will always be subtle differences. These differences become larger and larger as the instance of the activity becomes less and less familiar. To consider a more extreme example, if I am a guest in someone else's home, and decide to make tea, I follow the same general process that I do at home. The specific actions I perform however, will be entirely different from at home: from subtle differences such as where to place the cup when pouring water, to more significant differences such as

having to look through several cupboards to find the storage area for cups. This process is improvisation as well, just as certainly as the obvious large-scale substitutions described earlier. I follow my routine in these varying situations just as I do at home, and use my knowledge of tea-making and the concepts involved in making tea to replace, alter, or fill in gaps in my routine as required.

The ability to apply knowledge in this manner is at the heart of improvisation. It requires both the ability to use experience as contained in plans (i.e. the ability to deal with the "usual" course of events; in this example, to sharpen a pencil using a pencil sharpener), and also the ability to move beyond the usual course of action when necessary. The latter ability requires the agent to integrate its "usual" plan with knowledge of the situation in which the agent finds itself. For example, knowing that a cup is required to follow its routine, and knowing that such objects are usually stored in kitchen cabinets, the agent can go through cupboards in a strange locale when a cup is not immediately available.

Very rarely can we accomplish an everyday activity by following a known plan in explicit detail, or by ignoring plan knowledge entirely. The ability to apply a known method of accomplishing an activity in a specific situation requires the ability to apply plan knowledge in a flexible, opportunistic manner. This includes diverging from the previously successful routine at many points, integrating the circumstances of the situation in which the agent finds itself and the many additional types of knowledge outside of plans discussed in Chapter 2. Likewise, the ability to remember and apply what was done previously requires the structuring of plan knowledge, as does the ability to follow complex interacting goals. Plans are required, but must be used as *guides* to rational activity rather than definers of it.

This Chapter argues that the ideal method for an agent to accomplish everyday activities is through improvising on its routines, using compiled plan knowledge as a guide rather than as the sole means of providing knowledge for activity to the agent. I begin by examining how a plan can be used as a guide, and illustrating the range of behaviour to which this approach is well-suited. I then provide a formal definition of improvisation with respect to everyday activities, and describe in further detail how everyday activities may be accomplished by means of this process. The internal processing required of an improvising agent is described, and the requirements of a particular computational architecture are reviewed. Finally, an argument is made for a particular form of knowledge

representation in improvisation for everyday activities, leading to a concrete architecture presented in Chapter 5.

4.1. Using Plans as Guides

> Custom then is the great guide of human life.
>
> – David Hume, *Concerning Human Understanding*

I have used the term *plans as guides* several times already in this dissertation, without yet giving a precise definition. The reason for this is that following a plan can mean very different things in different situations. When one speaks of following a classical plan, it means using it as a computer uses a program: following each instruction explicitly [Agre and Chapman, 1989]. In the case of universal plans, I argue, it amounts to essentially the same thing. I have already argued (Chapter 2) that universal plans are simply caches of knowledge, compiled before an agent is placed in a given environment and indexed by situation for ready retrieval. The stepwise reaction of a universal planning agent is thus simply the result of choosing appropriate parts of a predefined "universal" plan. The agent is still directly following the plan, in that it cannot make use of any knowledge outside of the plan itself; it is simply choosing which pieces of it are dictated by the outside environment (that is, the "plan" is simply a very different structure from a classical plan). Agre and Chapman [1989] argue that this is using a plan as a guide; I argue that it is not, simply because the agent has no knowledge that is not compiled into the plan itself.

In fact, all plans, be they classical or universal, are simply caches of knowledge. Any plan is based on a large collection of knowledge: knowledge of the effects of action, of the behaviour of objects in the environment, the likelihood of success, etc. This is a vast collection, one to which humans refer constantly in order to guide them through their interactions with a complex and dynamic environment. However, this knowledge cannot simply be sorted through in a linear manner each time something is needed. Rather, we compile the most commonly used portions of all of this knowledge, and do things in ways that we know have worked before. Humans have plans and deal with things in routine ways simply to avoid having to look through this large collection of knowledge: we do things "without thinking", simply because in most cases there is far too little time to consciously realize the impact of each decision we make on every one of the many pieces of knowledge we possess. This is directly observable in many areas. I have already shown in Chapter 2, for example, that the purpose of recipes is to

remove the burden of extensive thought when cooking. This is the place of plans and plan knowledge in all everyday human activity: they serve as condensations or caches of knowledge, giving us ideas of what to do in particular situations or to accomplish particular desires, without the bother of having to go through every piece of our knowledge of activity and world around us.

This is precisely what a classical or universal planner does when it follows its planning algorithm. A classical planner considers all possible interactions between the world and the repertoire of actions at its disposal, and all possible restrictions that the organization of the world places on a solution to a problem, and constructs a sequence of actions to deal with a particular situation. It compiles and filters out all of the knowledge that lies behind the plan - all the reasoning involved in constructing it - and leaves a condensed sequence of actions. Similarly, a universal planner reduces all the knowledge necessary to perform in a given environment to a series of situation/action responses: *it filters out precisely the knowledge that one relies on to get around in the world in order to avoid having to consciously refer to that knowledge.*

This compiling and filtering of knowledge is a necessary part of being a resource-bounded agent in a complex world. Indeed, plans represent and exploit predictable aspects of the environment [Hayes-Roth, 1993], removing from the agent the responsibility of recognizing and making use of the predictable aspects over and over again. However, using a plan as a *guide* to activity means being able to refer to the large collection of knowledge from which the plan arises: to have it available when needed (to alter the plan and provide alternatives to it) and to be able to ignore it when it is not needed. Figure 4-1 illustrates this view of plans, using a classical plan for purposes of illustration[48]. The plan itself represents a condensed selection of the knowledge available for activity: that is, something that has been used many times before and can be relied upon to some degree. Associated with this plan, however, is the large collection of knowledge from which it has been compiled. Some of this knowledge is closely related to the plan, but not part

---

48 A universal plan would suffice equally well. Classical plans can obviously never be used exactly in any but one specific situation in a non-dynamic world. However, universal plans also suffer from this difficulty in practice, since there is no way to anticipate every possible conditional response in a given situation in any complex environment (as discussed in Chapter 3). No universal plan is truly universal.

of it. For example, in the pencil sharpening example in the previous Section, the role of the sharpener in the activity is not a part of a plan *per se*, but is an important piece of information if one wishes to improvise on the plan. Other knowledge is much more loosely associated with the plan. For example, the connections that allow some of the less obvious and more questionable substitutions described in the example to be made: knowledge of the purpose of the blade in the sharpener, and of other common kitchen objects and their relationship to the activity. Similarly, in making tea, my routine may tell me to go to a certain cupboard to obtain a cup, but I also have knowledge that this is only an appropriate default in my own home, and I have associated knowledge that supplies appropriate alternatives in other cases (e.g. asking where cups are stored, or searching cupboards in the kitchen).

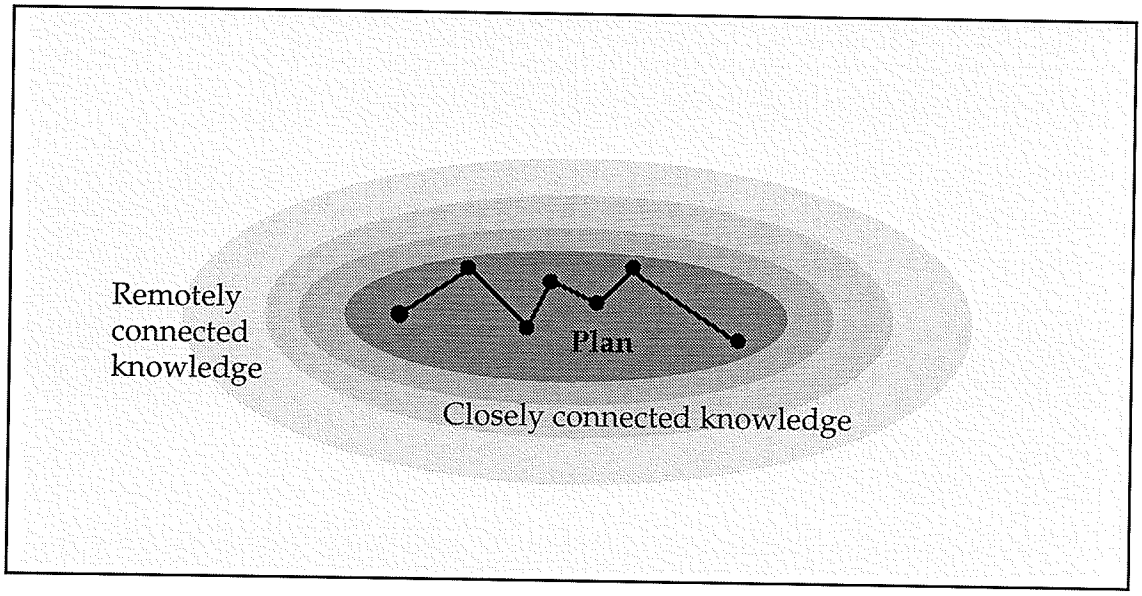

Figure 4-1. Plans as knowledge caches.

In fact, the condensed forms of knowledge that lie at the centre of Figure 4-1 need not even be classical or universal plans; they can be any form of condensed knowledge (for example, a piece of knowledge such as *prefer machine A to machine B* may subsume a large amount of knowledge that lies behind the preference). Such knowledge may be in the form of partial plans, skeletal plans, or complete plans; it may be in a form that would commonly call a plan, or it may not at all resemble such a structure. The point is that there exists both highly compiled and loosely associated forms of knowledge that can be applied as necessary. Since the associated knowledge is not directly

connected to the plan, much of it is common to many activities, and there is not necessarily a hard boundary between one such structure and another.

In a situation where plans can be followed directly, it makes good economic sense to do so, just as in a situation where a precise algorithm exists, it makes good sense to apply it as opposed to dealing with the problem heuristically. However, just as heuristic algorithms are needed for non-algorithmic problems, this collection of background knowledge is *absolutely* required when a plan cannot be followed in complete detail.

When applying a plan as a guide, the compiled plan knowledge is used as much as possible, for these obvious economic reasons. In a familiar situation, the more highly-compiled forms of knowledge may be followed very closely, without having to refer to the knowledge that lies behind it. In less obvious situations, the more loosely associated knowledge becomes relied on to a greater and greater degree. While some of this knowledge is in the form of plans, much of it is not. This is what it means to apply plan knowledge *as a guide*. The more closely the compiled forms of knowledge are being adhered to, the more closely one is following a plan. As these highly-compiled forms of knowledge are relied on less and less, the agent improvises more and more. When I speak of a *plan*, I therefore intend the term to encompass *any* type of highly-compiled knowledge of activity, regardless of the form of that knowledge. My use of the term *plan* thus encompasses both classical and universal plans, and a great deal more.

The structure shown in Figure 4-1, however, is much more than just a plan, even in the sense of this extended use of the term. Certainly, plans (in the sense of compiled knowledge, as described above) lie at the core of plans-as-guides, but as already mentioned, they fail to take into account the large collection of much more loosely connected knowledge on which they are based. To refer to the combination of highly compiled plan knowledge and the more loosely connected knowledge on which the former is based, I will use the term *routine*. This emphasises that plans-as-guides do not necessarily have a complete plan at their core, and that there is no hard and fast boundary where the core ends and associated knowledge begins. A routine is thus not a contiguous structure (in the sense of a connected series of steps), but a set of loosely- and tightly-connected knowledge for accomplishing some

activity[49]. Routines *are* plans-as-guides. The plan itself (the highly compiled knowledge at the core of the routine) serves as just one of many resources for activity. The internal and written recipes described in Chapter 2 are obvious examples of routines. They contain highly-compiled, often stepwise knowledge of how to prepare a given dish, but also require a great deal of background knowledge (some more obvious and more closely associated with the recipe than others) to be applied.

The ability to apply plans as guides is crucial for intelligent behaviour. While only now being acknowledged by AI, this fact has been long known in psychology. Birch [1945] for example, illustrates such behaviour in chimpanzees, and finds that the ability to select portions of contiguously learned sequences of behaviour (plans) and apply them in new sequences "makes possible an enormous expansion of the adjustive possibilities of an organism". That is, the ability to apply a plan as a loose guide to behaviour makes an agent more adaptive. At the same time, it maintains the goal-directed nature of behaviour, and therefore (by the definition given in Chapter 1) results in more intelligent behaviour.

4.2. The Range of Improvisation

In making use of both compiled knowledge (plans) and the background knowledge from which it is derived, improvisation is neither strictly planning nor strictly reaction, but applies to a very wide range of activities in between. Figure 4-2 shows the spectrum of activity described in Section 3.5 re-interpreted in terms of how strongly plan knowledge is relied upon. At the most basic level, an agent reacts spontaneously with no reference to plan knowledge whatsoever. This level explains undeliberated reactions, such as automatically dropping a hot cup. Above the level of spontaneous reactions is the complete following of detailed plans. This represents an agent's experience (in the form of compiled plans) completely and directly controlling activity, with no outside resources: for example, operating a machine on an assembly line.

---

[49] Chapman [1990] was the first to use the term *routine* to emphasize the difference in structure between plans and other forms of knowledge that support activity. He, however, uses the term to mean "a regular, patterned practice of interaction", which is "typically not represented by agents". This is not a knowledge resource, as I intend its use here, but a universal-planning interaction with the world, describing the low-level forms of activity shown in Figure 3-7. This will be explained further in the next Section.

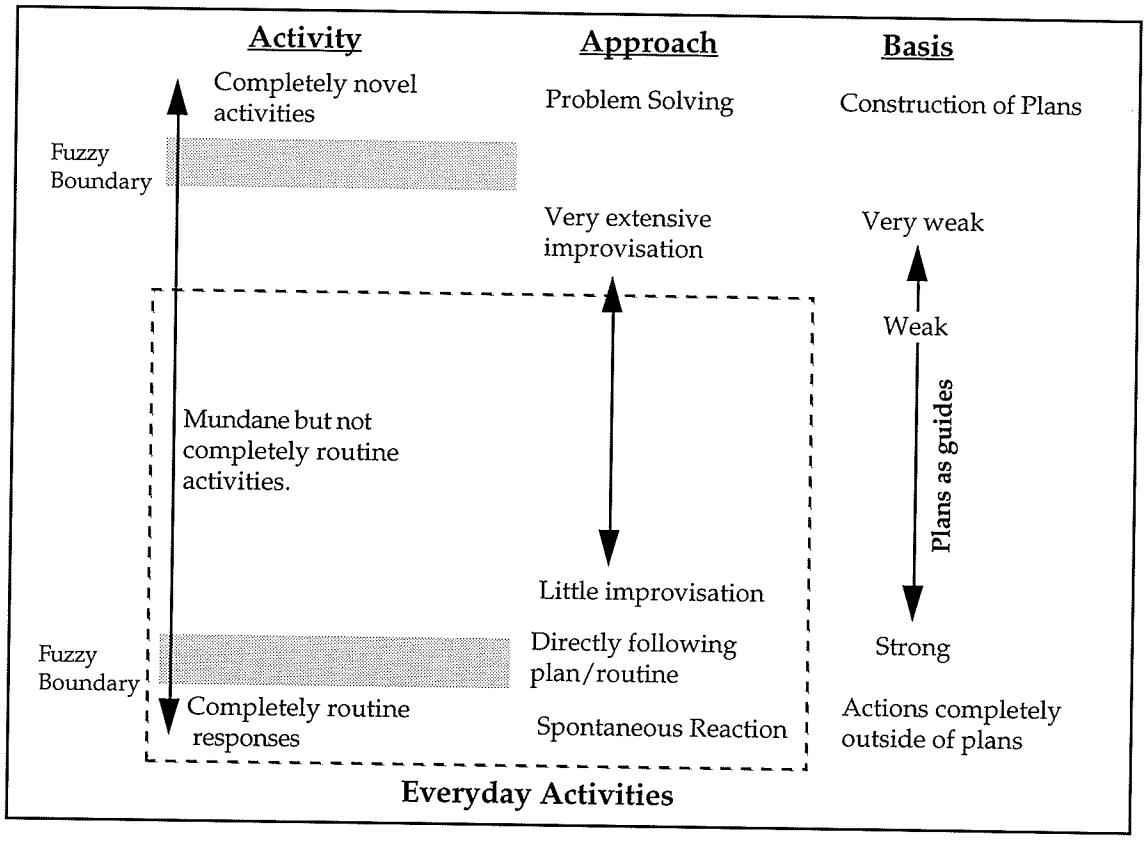

Figure 4-2. Viewing the spectrum of activity in terms of improvisation.

Above these levels, the knowledge that guides activity becomes less strictly reliant on compiled plans: we begin to move away from the condensed, compiled knowledge that form plans, and make greater use of knowledge outside of plans. This is weak improvisation, where experiential knowledge (much of it in the form of highly compiled plan structures) still guides very strongly, but must be diverged from on occasion as circumstances warrant. An example of such weak improvisation would be the case of making tea in a very familiar situation. The agent may have always performed essentially the same sequence of actions, using the same tools and techniques, and makes use of this routine to guide it through the activity. The agent does not (and cannot) *rely* on the highly-compiled aspects of this routine in the way that classical planners rely on detailed plans. The difference is that the agent has the power to diverge from the most obvious parts of the routine at any time - it uses the core of the routine (plan knowledge) for intellectual convenience, and deeper reasoning (the periphery of the routine) to provide alternatives when necessary. Indeed, the agent must often diverge from what might be

considered a routine series of steps for tea making, even in a very familiar situation: cups drop, we run out of tea, and kettles boil dry.

This deeper reasoning also comes into play when the situation diverges from that with which the agent has the greatest amount of knowledge. For example, if an agent were to make tea in someone else's house, the same processes would be involved. However, the agent can no longer rely on much of the compiled knowledge on which its home tea routine is based: the locations of objects, or even whether or not they are available, for example; or whether the home in which the agent finds itself uses a stove or an electric kettle to make tea. Here, the knowledge outside of the compiled parts of the routine becomes crucial. The agent's general knowledge of cups for example, may include the fact that they are usually stored in a cupboard in the kitchen, allowing the alternative to begin looking for one there. More extensive improvisation requires the agent to make greater use of the knowledge that forms the background of its everyday routines (that is, knowledge outside of the plan-like structures that suffice for simpler forms of activity) to perform activities in less-than-normal situations.

This knowledge is also required where the compiled knowledge at the core of the agent's routine is not strong enough to provide obvious alternatives for action. For example, if the agent does not have a specific method it always follows when making tea (i.e. it doesn't make tea very often), compiled knowledge in the routine may simply specify that boiling water is necessary, with no obvious preference. In this case it is the agent's background knowledge of how water is to be boiled that must be used to provide the necessary actions for the agent to carry out. Here the compiled core of the routine is acting as a guide, but weakly. It directs the agent to the appropriate knowledge to apply to perform the activity rather than specifying the appropriate alternatives itself.

As shown in Figure 4-2 (and as is intuitive from these examples), improvisation occupies a wide range of the spectrum of human activity. The size of this range is also supported by the few other studies of the range of human behaviour that have been made to date (e.g. [Hayes-Roth, 1993]). Indeed, it is possible to extend the principles of improvisation to the point where the basis in experience is so weak that the improvisation process becomes indistinguishable from problem-solving. No doubt the processes at this level are interesting in that they provide a cross-over from improvisation, through case-based planning, to problem-solving; however, long before this level is reached, the application to everyday activities is left behind.

4.3. Defining Improvisation

Having given a basic overview of what improvisation is expected to explain, we must now work toward a more formal definition of the concept. Despite being able to give many examples of what I consider improvised behaviour to be, improvisation itself is not an easy term to define. Jencks and Silver [1972] define improvisation (which they also term *adhocism*) as a general and loose approach to a problem as opposed to a strict and systematic one. This does capture some of what we commonly think of improvised behaviour, but is not particularly useful from a computational point of view. The main difficulty with a definition from this point of view is that improvisation is not often thought of in the information-processing terms that have been used thus far. The term is most commonly associated with the theatre, and indeed, this is one of the few areas where improvisation as a mechanism for performing an activity is intensively studied. Hodgson and Richards [1974], in their study of theatrical improvisation, describe three common senses of the term:

- The process of "making do", or coping at a basic level with some activity using minimal resources. For example, temporarily replacing a broken car fan belt using nylon stockings.

- The process of producing quality results using inferior materials: for example, turning old discarded clothing into new fashions, or producing a gourmet meal using leftovers.

- The process of adjusting to the occurrences around oneself while working at a particular activity; of being receptive to the world around oneself, and using knowledge of that world to adjust to adapt to change.

All three aspects are part of the colloquial concept of improvisation, and all three contribute to the process of performing everyday activities by flexibly applying knowledge of experience to cope with a dynamic world. The latter captures this most obviously. However, the first two capture aspects of improvisation that are equally important from an AI perspective. As discussed in Chapter 2, the process of carrying out everyday activities is the epitome of satisficing: an agent uses the limited experience and resources at its disposal to carry out an activity in a (usually) time-bounded situation. The first sense of improvisation described above encompasses this, but also has a connotation of being inevitably imperfect and often inadequate. This connotation of imperfection is fairly strongly associated with colloquial

improvisation; that of inadequacy much less so. An improvised method for accomplishing an activity (that is, one based in flexibly applied, limited experience) will never be as good as one where the agent has enough experience to rely on a direct plan. As will be discussed shortly, it is an inherently satisficing method of accomplishing an activity. The second of Hodgson and Richards' aspects of improvisation is also related to this: should an agent have extensive knowledge of an activity, it should be possible to produce results that are better than might be expected from an agent with less experience, even when using inferior materials. This depends solely on the circumstances around the activity and the quality of other resources available to support improvisation[50].

Combining the above aspects, we arrive at a definition of improvisation from the perspective of the intelligent performance of everyday activities:

> *An agent improvises in the performance of an everyday activity when it possesses compiled knowledge describing its routine method of going about the activity, and uses associated background knowledge of the activity and the world around itself to apply its routine flexibly in varying situations, in order to accomplish the activity in a satisficing manner.*

The term *improvisation* is not without previous use in AI. Most of these uses have been in its colloquial sense. Of the few that go beyond this level, most prominent is Agre's [1988; Agre and Horswill, 1992] use of the term to denote universal planning activity. Agre makes extensive use of the term, and defines improvisation as *continually redeciding what to do* [Agre, 1988]. This captures part of what improvisation is from the perspective of everyday activities, in that it differentiates it from creation of detailed plans and emphasizes that it involves the agent adapting to circumstances dictated by the world around itself. Indeed, deciding what action to take on a continuous basis is an important part of improvisation in everyday activities. However,

---

50 There is also an aspect of creativity associated with this sense of improvisation. It is not the claim of this dissertation that improvisation can be used to explain creativity, an area long of interest in AI. Such a claim would require work far beyond that presented here. However, this techniques of improvisation described in this Chapter and Chapter 5 show strong connections to creative behaviour. Indeed, the act of flexibly applying routines using general background knowledge seems to be an indication of creativity. The application of improvisation to creative behaviour is is a promising avenue of future research.

Agre's definition fails to encompass the satisficing element in the very nature of both improvisation and everyday activities. More significantly, it fails to encompass the fact that there must be some inherent *basis* for improvisation: we improvise during the course of everyday activities not simply by reacting to the world continuously, but on the basis of the agent's previous experience, using more general knowledge to provide alternatives when the usual routine is not suitable.

Outside of the realm of everyday activities, a technique with some similarity in spirit to what I have defined as improvisation has been demonstrated in Wilkins' [1980] use of plans in chess. Wilkins' *Paradise* system used planning techniques to produce plans that represent possible moves on a given board position (e.g. a trapped piece might direct the system to produce a plan to capture that piece). The plan is then used as a guide for a search for the best possible move. If a plan can be followed (i.e. matches a given set of preconditions) it is used as opposed to searching individual moves. If, during the execution of a plan, a number of other plans are found to apply, they are evaluated, and the original plan may be abandoned. The use of plans to guide search greatly reduced the number of possible board states that must be examined: to tens and hundreds from thousands to hundreds of thousands [Wilkins, 1980]. This is a form of improvisation, in that both plan knowledge and knowledge outside of plans are used to carry out activities. However, it is simplistic: plans are simply followed or not followed, they are not diverged from except for complete abandonment. In addition, while chess is combinatorially complex, it differs extensively from everyday activities: it requires much more optimal performance and uses many problem-solving techniques that are completely foreign to everyday activities - it represents the upper bound of the spectrum in Figure 4-2. Chess is also relatively simplistic in comparison to everyday activities, in that there is a much smaller number of concepts that must be known in order to function, and comparatively few alternatives from which to choose at any point in time.

4.4. Improvisation and Everyday Activities

The mention of satisficing in the definition of improvisation given in the previous Section is critical. As already mentioned, improvisation is a very heuristic process, in that the agent does the best it can given bounds on its resources and cannot guarantee the quality of its results. It is therefore unacceptable in any domain where satisficing is not tolerable, explaining why improvisation is commonly connected with theatre, home car repairs, or storytelling, rather than law, medicine, or accounting. This is by no means a

black and white issue. Rather, it is a question of the *degree* of improvisation that is tolerable in any domain. Even in very structured domains where high quality results are demanded, improvisation is performed to some degree. Wilkins [1980], for example, demonstrates that adhering too closely to a detailed plan can be detrimental even in a very structured domain such as chess, while outside of AI, Minzberg [1971] has shown that managers often make decisions using informal reasoning and intuition, and only rarely make use of explicit, strict decision-theoretic models. Even in domains such as law and medicine, improvisation can be seen to some degree.

In addition to requiring a domain where satisficing is tolerated, the domain must be inherently heuristic for improvisation on any significant scale to make sense economically. As described earlier, if an exact plan for solving a given problem is available, it makes no economic sense to use the heuristic techniques of improvisation, just as there is no need for heuristic method when confronted with a problem when a known algorithm exists. Heuristic methods are almost always more expensive computationally than any algorithm, and improvisation is no exception. While it can have a firm basis in a plan, the ability to apply knowledge outside of that plan is certainly more computationally expensive than using the plan directly.

I do not argue with the fact that improvisation is an approach whose results are usually not optimal, nor do I claim that it is an appropriate or even adequate approach for activities that do not tolerate satisficing well. Indeed, improvisation is an approach that shows all the weakness of human reasoning: it may never find a solution, and certainly will rarely find the best one; it cannot guarantee the amount of time it will take, or even that its solution will always improve with more time. The previous Section has illustrated a spectrum showing that there are many activities where improvisation is not required for activities. What benefit, then, is such an approach to AI? Why would we possibly want to introduce these human frailties to activity?

The answer to these questions is to deal with the types of activities to which these frailties are not significant, and may even be considered strengths: *everyday activities*. I have shown in Chapters 2 and 3 that everyday activities rely on the flexible application of experience, taking advantage of as much structure as is inherent in the activity, and performing in a satisficing manner. These are exactly the qualities that improvisation provides. While the approach offers no guarantees, it more than makes up for in flexibility. It is this inherent flexibility that allows improvisation to accomplish such a broad spectrum of behaviour, from closely following plans, to barely being

influenced by them. Jencks and Silver [1972, p. 16] describe this flexibility particularly well:

> The characteristic *ad hoc* amalgamation contains much that is inessential, much that is fortuitous and redundant. But if it is not as refined and precise as other kinds of purposeful action, then as least it is more open, suggestive, and rich in possibilities. The extraneous material suggests new uses, whereas the perfected and refined construction is usually confined to its specific ends.

While there are certainly activities that do not require improvisation as I have defined it (e.g. highly abstract problem-solving, simple reaction), the bulk of human activities, and all of what I have termed everyday activities fall in the spectrum covered by improvisation (Figure 4-2). Improvisation is inadequate from the perspective of many activities, but not everyday activities. For these, improvisation is a necessity. It is a robust, flexible approach to dealing with tasks whose complexity cannot be dealt with in any other manner. *It is the nature of the tasks involved, rather than the nature of the approach that dictates the characteristics of improvisation.* Improvisation is simply the methodology humans have developed to cope with activities with the characteristics described in Chapter 2 and summarized in Section 2.5. It is the challenge of AI, in attempting to cope with everyday activities, to accomplish those characteristics within a computational architecture.

In Chapter 2, I argued that humans perform everyday activities easily in spite of their complexity by taking advantage of the significant problem structure exhibited by its previous experience with the everyday activity (Figure 2-2). This improvisation process can be viewed as taking advantage of as much structure as is inherently available in the problem, and is greatly underconstrained when compared to the amount of structure necessary for classical and universal planning.

Consider Figure 4-3, which illustrates the problem structure exploited by a classical plan. In this case, complete knowledge of the problem structure must be available beforehand and compiled in the form of a classical plan. When this plan is then executed, this compiled knowledge is followed exactly. The reason that such plans are inapplicable to most everyday environments is that they make such overconstraining assumptions about knowledge of the problem structure.

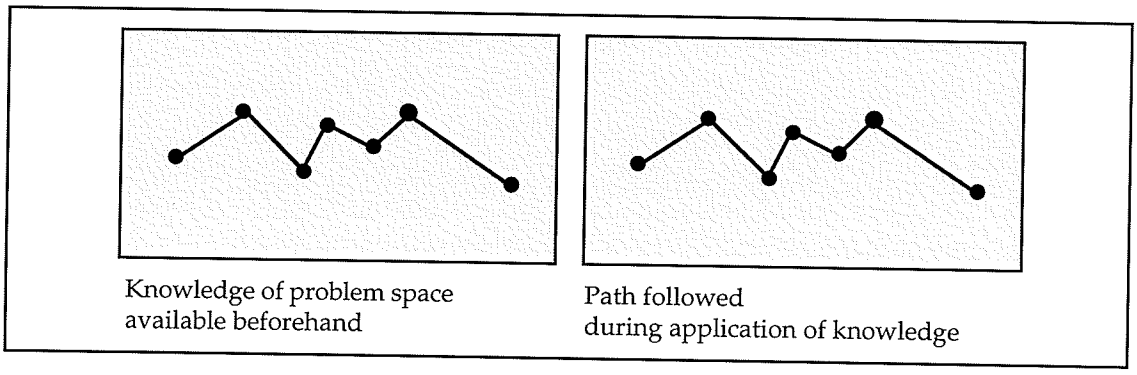

Figure 4-3. Structure exploited during execution of a classical plan.

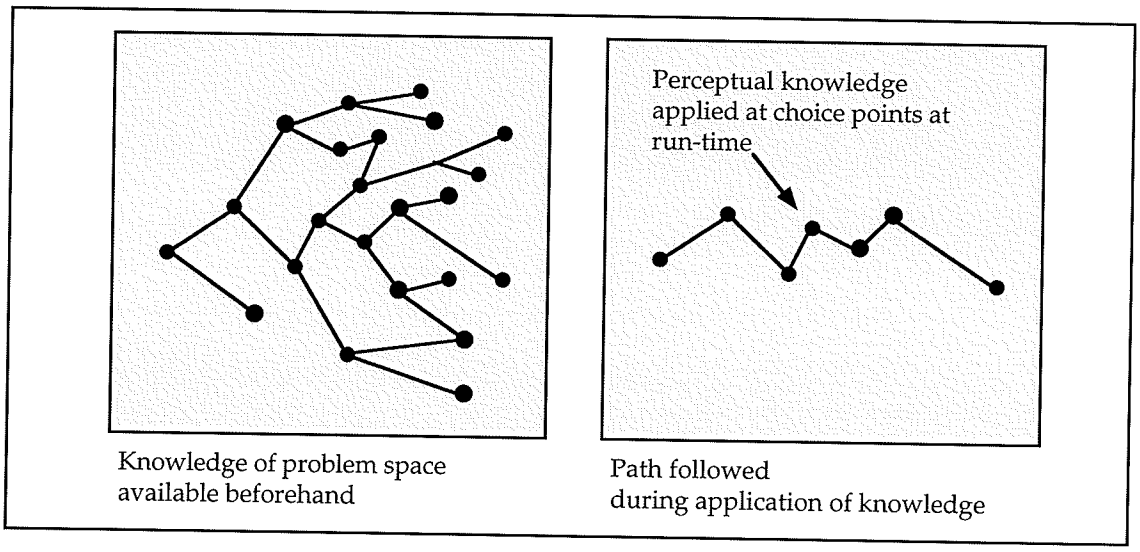

Figure 4-4. Structure exploited during execution of a universal plan.

Figure 4-4 shows the problem structure exploited during the execution of a conditional or universal plan. Recall from Chapter 3 that a conditional plan is a classical plan with possibly multiple branches at any step, computed ahead of time (as is the rest of the plan) and attempting to control for uncertainty at decision points. These are essentially small universal plans, since a universal plan simply assumes it has all possibilities computed for all choice points. After this structure is computed, it is applied in in a given world, and perceptual information at each choice point allows the agent to select the pre-computed portion that applies in a particular situation. Figure 4-4 shows that conditional/universal plans also make overconstraining assumptions of the problem structure: they assume, like classical plans, that this structure is completely available to them ahead of time, and use perceptual information

at run time to choose the appropriate path. The routines used by improvisation, in comparison to those of classical and universal planning, take advantage of only as much structure in the environment as the agent knows about, and also only as much as the resources expended on the problem allow.

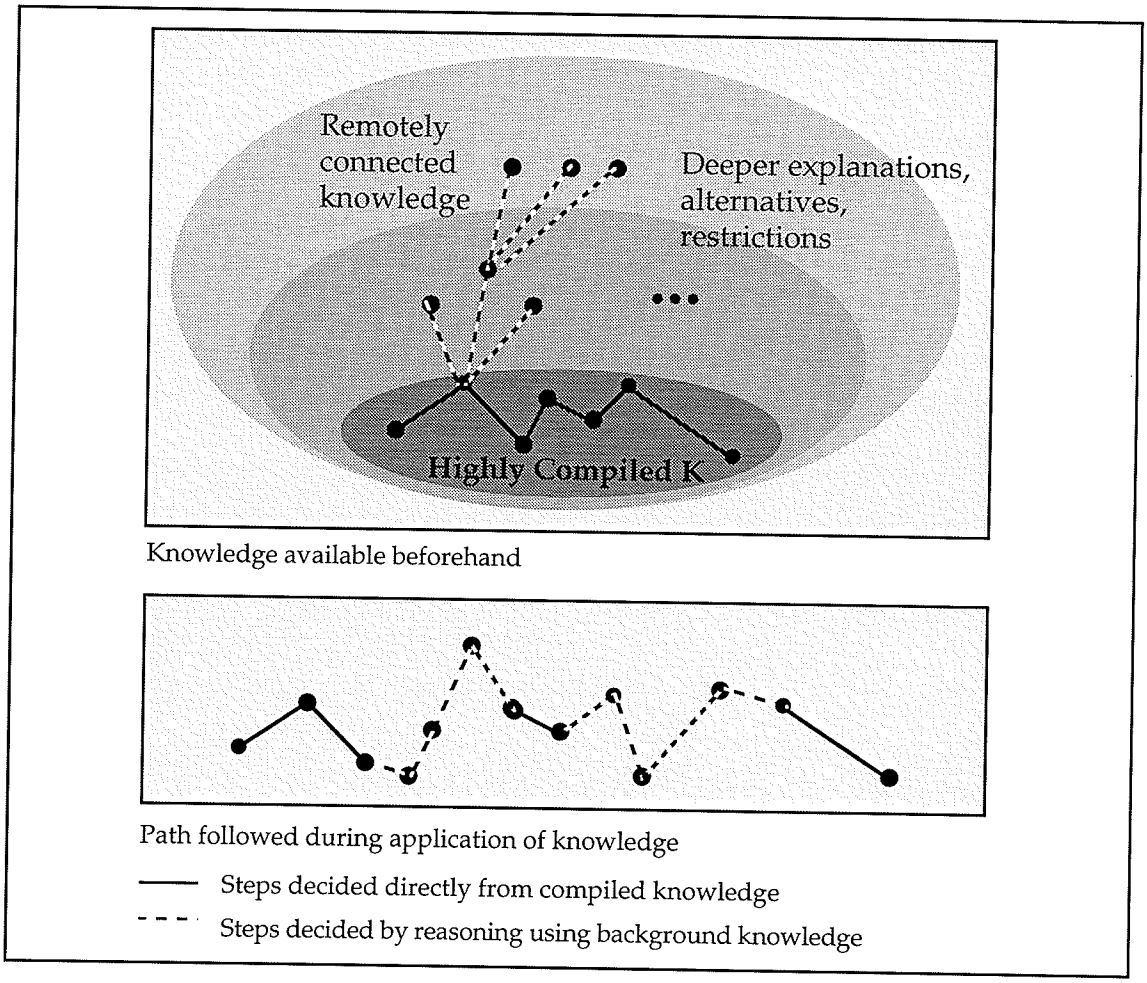

Figure 4-5. Problem structure exploited by improvisation.

Figure 4-5 shows both the knowledge available to an improvising agent performing some activity and the results of applying that knowledge in a particular situation. As previously described, this knowledge consists of both highly-compiled plan knowledge, and the loosely associated background knowledge from which the compilation was performed. While the routine guides the agent through the activity, it is applied flexibly. To as large an extent as is possible, the compiled knowledge is relied upon. While the

background knowledge encompasses everything the compiled knowledge constraints, it is not consulted in most situations simply because the compiled routine is enough (for the economic reasons detailed in Section 4.1). In situations where the compiled knowledge is inapplicable (e.g. when something goes wrong, or when the situation in which the agent finds itself does not precisely match that it is used to), the background knowledge associated with the compiled knowledge can be accessed as necessary.

The lower half of Figure 4-5 illustrates an example. The agent performs a series of actions, some of which come directly from the core of its its routine (compiled plan knowledge). Other actions, however, arise from deliberation using the background knowledge associated with the plan at the centre of the agent's routine. This could be because a portion of that plan was inapplicable in the particular environment; because it had to divert from the plan to accomplish some activity the plan did not include; because it was integrating activity with other routines to satisfy other desires; or because the background knowledge simply indicated that other alternatives made more sense than what the plan recommended.

In this example, the agent returns to portions of the core of its routine at intervals. In another situation, the agent might follow the compiled portions of its routine more closely, or alternatively, the first action it took outside its compiled routine might invalidate conditions behind the routine itself, forcing abandonment. In general, It may be possible that the situation in which the agent finds itself precisely matches the compiled portions of the routine it is used to following, and none of the agent's extensive background knowledge might be used at all. This situation could also arise when there is not time to consult the agent's background knowledge. It may also be possible that the plan will be only minor help; in such a case, background knowledge might be applied extensively. This is what is meant by *taking advantage of as much structure as is inherent in the activity.* In some cases the agent may know a great deal, and may rely on heavily compiled knowledge. In other cases, the agent may know very little applicable to the immediate situation, and may rely heavily on deeper reasoning.

Overall, the relationship of improvisation to classical and universal planning is akin to the relationship between heuristic search and non-heuristic search. Simon [1969, p. 35] observes:

> The technique of heuristic search makes much weaker demands on the problem structure than do linear programming or linear decision rules, but it can generally find only satisfactory solutions, not the optimal one. On the other side of the coin,

heuristic search can handle combinatorial problems (e.g. factory scheduling problems) that are too large for exact solution even with the biggest computers.

Like heuristic search to non-heuristic techniques, improvisation makes much weaker demands on the structure of the problem than does classical or universal planning. Improvisation quite literally "makes do" with whatever knowledge is available and whatever time is available for it to access that knowledge.

4.5. The Improvisation Process

To this point, much has been said about the nature of improvisation and the role that improvisation has to play in everyday activities, while saying little about the internal computational processes that are required when improvisation is performed during the course of everyday activities. This Section describes improvisation from a computational perspective, focussing on the required processes and knowledge for an improvisational architecture for everyday activities at a general level, rather than advocating any one particular architectural design. An architecture embodying the principles discussed in this Chapter will be described in Chapter 5.

Figure 4-6 indicates the relationship of the four major components of any improvising agent:

- *Memory*. As improvisation is based in experience, memory is the most vital component of an improvisational agent. Memory must be organized in terms of reminders (cues); a specific desire and a specific external situation should remind the agent of a particular routine it has used in the past (e.g. making tea at home). They may also remind the agent of a specific reaction to the situation at hand (as a universal planner would). This is not the only type of reminder that must be supported by this memory. An improvising agent relies on much knowledge that is connected not necessarily with a specific external situation or with the core of a particular routine, but instead is connected to concepts describing its understanding of the world around it (i.e. background knowledge in routines). For example, if the agent comes across a container of milk sitting on the counter, its concept of milk should remind it that milk should be kept in the refrigerator. Such reminders may be triggered by something in the external environment (as in this example), or a combination of environment and the routine the agent is following.

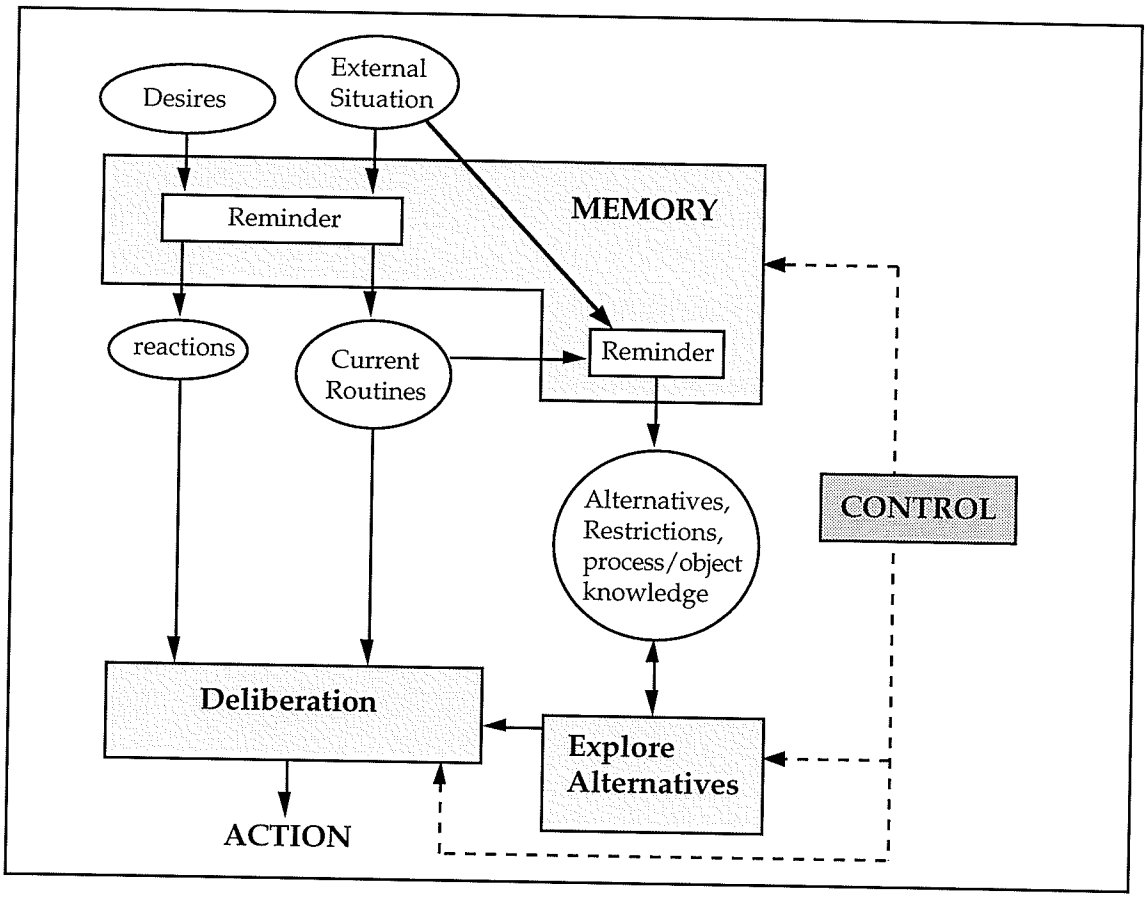

Figure 4-6. Abstract view of the mechanisms comprising improvisation.

- *Exploration.* As mentioned above, many alternatives will arise to the actions supplied by a routine during the course of participating in an everyday activity. Some may arise directly from compiled knowledge, while others require a more active exploration of background knowledge. This is most obvious when routines go awry. For example, if I am making tea, the the tea bag breaks, a number of alternatives arise, each along the lines of somehow getting the tea leaves out of the tea or starting again. In the former, there are many ways of doing this, just as there are many ways of sharpening a pencil (Section 4.0). This process allows such alternatives to planned activity (or alternatives where no plan suffices directly) to be explored.

- *Deliberation.* As alternatives for action become available (from the compiled or background knowledge in routines), an agent must have a decision-making process that selects the action (or actions, if parallel

actions are supported) that the agent will perform. This deliberation may be extensive, or a choice may be obvious. The amount of deliberation necessary is dictated by the activity the agent is participating in, and the world around it (e.g. there may be an obvious choice to get out of the way of a moving truck, or a more sophisticated weighing of alternatives in a less obvious or time-constrained situation).

- *Control.* Any routine will be associated with an extremely large set of background knowledge, some very tightly connected (i.e. highly compiled), some more loosely (as per Figure 4-1). This means that for any routine the agent is following, the routine may generate a combinatorial number of reminders of additional pieces of knowledge, and the exploration process a similar number of alternatives. The deliberation process can spend an arbitrary amount of time deliberating any decision, and the memory, which presumably has a finite bandwidth and speed, must access the most needed knowledge first, even when many requests are available. A control module controls the operation of each of the above processes in order to have the resource-bounded agent operate in a satisficing manner.

Improvisation operates essentially through reminders (cues). An unsatisfied desire, together with the external situation in which the agent finds itself and the intentions for activity it has already adopted, reminds the agent of a routine to satisfy that desire, given the situation. The recognition of a specific set of external circumstances may also trigger a reaction to the recognized stimuli.[51] These represent obvious possibilities for action. Since a routine is not a complete plan, many of its recommendations for action may not be possible, and there may be many reminders of additional possibilities for action that arise from the background knowledge associated with the routine (e.g. my routine for making tea at home does not immediately involve anything to do with milk, except that my tea needs some in it. Reminders are necessary to get the milk and to put it away). These reminders are

---

51 As described in Section 4.2, strict reaction falls outside the boundaries of what I have defined as improvisation, but is necessary to support everyday activities. Actions such as dropping a hot cup are certainly not part of any routine, and indeed, are rarely consciously recalled. Examples such as this are the limit of reaction in this architecture, however. I have already argued (Chapter 3) that the Universal Planning model as proposed by [Schoppers, 1987; Agre 1988] due to the requirement of a reaction for every possible situation the resource bounded nature of an agent. The nature of reactions in improvisation is discussed further in Chapter 5.

accomplished through actively exploring the background knowledge behind a routine. This exploration may be as trivial or extensive as time allows. The various alternatives are deliberated, and action(s) selected for performance. This will change both the internal state of the agent (the intentions and expectations it has for the future) and the external state of the world, and new activity will arise from these alterations. From this process, coherent behaviour in the midst of complexity and uncertainty emerges. This process of flexibly applying the agent's previous experience captures the essence of Bartlett's [1967] view of activity quoted in Section 3.2.1: "When I make the [tennis] stroke I do not, as a matter of fact, produce something completely new, and I never merely repeat something old".

These processes are dictated by the requirements for everyday activity laid out in Chapter 3 and the characteristics of everyday activity described in Chapter 2. An agent has compiled knowledge in its routines that can be used as a resource for activity, and also possesses the ability to apply more loosely connected, "deeper" knowledge to alter the agent's routine and also act spontaneously outside of the routine. The agent's intentions for the future (that is, the routines it intends to follow) also evolve continually over time: since the routine does not exist as a single contiguous plan, parts of a routine may be created dynamically or substituted from other active or inactive routines.

The actual process is much more complex than this: as Figure 4-6 indicates, any one desire is part of many that are active at any given time, just as any routine is part of a collection the agent is following[52]. Figure 4-6 also omits many things that must be part of any particular improvisational architecture: perception, and maintenance of expectations of the world, for example. Figure 4-6 is meant only to be an illustration of the *general* components required in an improvising agent. A specific architecture, including knowledge representation mechanisms that allow the agent's routines to evolve during use as detailed above will be described in the next Chapter.

---

52 Recall from Chapter 2 that everyday activities overlap in both subsumption and interactive relationships; routines must interact in a similar manner.

4.5.1. Improvisation as Intelligent Control

The processes involved in improvisation during everyday activities are inherently different from what is normally considered planning in AI. A planning problem is traditionally defined as one that is solved by the construction of a sequence of actions (a plan). Improvisation in everyday activities both follows and eludes this definition at the same time, because it involves the *application* of plans, rather than their construction. Indeed, by virtue of the fact that the agent is performing everyday activities, extensive experience is relied upon, and very little in terms of planning *per se* should be involved. In making decisions to carry out actions, an improvising agent is evaluating which pieces of experience-related knowledge apply most strongly to a given situation: this has little to do with constructing sequences of actions. On the other hand, in doing so, the agent may be integrating parts of many general plans. In many cases, the routine that an agent follows does not even exist as one contiguous structure (i.e. the core of Figure 4-1, or as one branch of a universal plan), but rather are only very loosely associated with one another, and are built up piece by piece as the activity unfolds. While not exactly planning, this process certainly bears some similarities to the construction of plans. Furthermore, deciding rationally what to do at one particular moment involves considering future commitments and intentions (Section 2.4), again fitting planning behaviour. The difference between improvisation and planning is that in improvisation, the agent is essentially planning as it goes along, rather than planning ahead.

In recent years, many forms of dealing with activity have arisen that have some basis in the planning paradigm, but have not arisen directly from it (universal planning, or case-based planning, for example, have aspects that bear some allusion to planning and aspects that are very distantly removed from it). As mentioned in Chapter 1, this has caused a stretching of the boundaries of the field, to the point where most modern definitions of the field are of the form "using some type of construct that constrains or provides guidance for the actions of an intelligent agent" [McDermott, 1991]. Others (e.g. [Agre, 1988; Chapman, 1990]) would argue that such forms of planning are not planning at all, and should be strongly differentiated.

There is a great basis in this argument, if only because pure planning has been so strongly unsuccessful in everyday activities. Because of this, I argue that improvisation is more properly termed a problem of *intelligent control*: of deciding what action to perform at a given time, in light of the world around the agent and its intentions for the future. When an improvising agent

carries out a routine, it is not because that routine exists as one contiguous structure in its memory (as a plan would). Rather, a routine is built up piece by piece, with each piece being followed (or the agent even being reminded of its existence) because the situation warrants.

4.6. Toward an Architecture for Improvisation

The previous Section serves as an introduction to the computational processes involved in improvisation. Any particular architecture for improvisation, however, must answer many questions raised by this introduction. Not only must it include aspects that are glossed over in the previous Section (e.g. maintaining expectations of the world, handling perception), it must deal with many basic concerns of the major components of improvisation. These concerns are many, but can be grouped into two basic areas: an architecture for improvisation must develop a specific structuring of knowledge to support the processes described above; and it must also develop a specific control methodology in order to control the inference processes involved. These will be dealt with separately.

4.6.1. Knowledge Structuring in Improvisation

As described in Section 4.4, the routines that are used in the process of improvisation allow the agent to take advantage of precisely the amount of structure that is inherent in the activity it is performing. These routines are extremely flexible compared to classical and universal plans. The flexibility inherent in routines can be divided into three areas:

- The guidelines provided by the plan knowledge within a routine may be strong or weak; the agent can choose to follow any action the routine recommends or ignore it. This allows a routine to be flexibly integrated with others the agent might be following.

- The range of highly compiled (surface) knowledge to much more loosely structured background (deep) knowledge allows the agent to rely on plan knowledge to a high degree in familiar situations, and to make use of background knowledge when plan knowledge is unreliable. This allows the routine to be useful in a much wider range of situations than a strict plan. It can also allow much more accurate performance of an activity when more resources (time and knowledge) are available.

- Because plan knowledge is integrated with the deeper knowledge that forms its basis, a specific instance of a routine can adapt "automatically" to many situations as they arise, simply because the considerations for these situations are part of the routine itself.

The latter point requires further explanation. If one were to use a classical plan as a resource for a particular activity (say, shopping), and then imposed further conditions on that activity (such as being in a hurry, or having very little money), the entire plan would have to be changed. This is because during the creation of the plan, assumptions are made that allow the separation of the compiled knowledge in the plan from other knowledge the agent may possess. In this case, a given degree of hurry and a given amount of money is assumed. If these change, there is no link from the portions of the plan affected by this to the original knowledge used to generate those parts of the plan. Thus while saving time, the plan becomes monolithic.

In the case of a routine for grocery shopping, while there may be a particular and fairly precise sequence of actions the agent follows in its regular shopping, the knowledge behind this sequence is also available, and when there is little money or more hurry, this background knowledge is immediately available. The parts of the routine that are incompatible with these aspects are ignored, and new portions of the routine may be performed. The flexibility is inherent in the maintenance of both the compiled and distributed forms of knowledge. Different portions of one routine are active in different situations, rather than using completely separate plans. This is very different from choosing various branches of a universal plan in different circumstances. A given set of circumstances is allowing the routine to be modified dynamically: some parts become more active, others less so. It is not a pre-compiled branch that is selected over others, but rather the interaction the compiled parts of the agent's routine and knowledge external to it, allowing actions to be selected dynamically over and above what is specified in that compiled knowledge.

The major question that any particular architecture for improvisation must answer is how this knowledge is to be structured. An example yields some insights. Consider Figure 4-7, which shows a complete series of actions for a specific instance of an improvised tea-making activity. It is the specific sequence of actions I followed one particular time I made tea in my own home. Some of the actions were actions I always follow when making tea, while others were new to this particular instance. This is not the description I would give if I were asked how I make tea, and it certainly doesn't resemble the internal tea recipe described in Chapter 2. However, it represents the

output of improvisation: a specific sequence of actions derived as the activity is ongoing, and accomplishing my desire. Many of the actions are not what one would think of as part of tea-making (e.g. putting milk away, closing refrigerator doors). Rather, they are interactions between the way I make tea and the way I do other things (in this case, keeping the kitchen neat).

1. go to stove
2. pick up lid of kettle, check for water in kettle
3. replace kettle lid
4. set stove element control to "high" mark
5. go to cupboard
6. open cupboard door
7. pick up a cup
8. close cupboard door
9. go to counter by stove
10. put cup on counter by stove
11. pick up lid from tea canister
12. pick up a tea bag
13. put bag in cup
14. wait for water to boil
15. turn stove off
16. pick up kettle
17. fill cup with water
18. replace kettle on stove
19. open drawer
20. pick up spoon from drawer
21. close drawer
22. squeeze tea bag with spoon
23. pick up tea bag with spoon
24. go sink
25. open cupboard under sink
27. close cupboard door
28. place spoon in sink
29. go fridge
30. open fridge door
31. pick up milk from fridge
32. close fridge door
33. go counter by stove
34. open milk
35. pour milk in tea
36. close milk
37. go fridge
38. open door
39. place milk in fridge
40. close door
41. go counter by stove
42. pick up cup
43. go table
44. sit down
45. drink tea

Figure 4-7. A particular instance of improvised tea-making.

It would be exceptionally difficult computationally to apply this structure during improvisation. If such a plan-like structure were used as a basis for improvisation, a great deal of analysis would have to be performed during the process to understand why certain aspects (e.g. action 22) were in the plan, and what in the environment would indicate whether it made sense to perform that action. A failure in the plan early on would quickly invalidate

the rest of the plan, and extensive symbol manipulation would be required to repair it[53]. The amount of time such processing would require would almost certainly always be beyond the required response time for improvisation. We would, in every sense of the term, be back to classical planning.

A structure such as that shown in Figure 4-7 is far too detailed for use in many different situations. However, the sequence of actions is clearly the result of the integration of one or more routines, since it is the output of the improvisation process. All the knowledge displayed in Figure 4-7 must be represented within the agent at some point (either in plan or background knowledge). The question that must be answered by an architecture for improvisation is, at what level should this knowledge be represented? Clearly, a much more abstract level is necessary in order to be able to perform improvisation. However, consider the complications at this abstract level. Getting from an abstract description such as the internal tea recipe shown in Figure 2-3 to a detailed sequence such as that shown in Figure 4-7 would also seem to be a very intricate process: once again, far too complicated for something as quickly and easily performed as everyday activities. Somehow, an improvising agent can go almost directly to the detailed actions it needs when it is in a recognizable situation, with no intellectual analysis whatsoever of the actions it is performing, and can also automatically move to background knowledge when necessary.

This results in a paradox: on one hand, beginning with the background or abstract methods of doing things and moving to specific actions from these violates much of what is observable about everyday activities. On the other hand, it is computationally unfeasible to begin with a specific routine and rely on background knowledge when that routine fails. Somehow, both the highly compiled plan aspects (the obvious actions in a particular situation; many of the actions in Figure 4-7) are naturally and seamlessly integrated with the use of background knowledge when humans improvise during the course of everyday activities. Any specific architecture for improvisation must explain this paradox.

---

53 [Hammond, 1989a] illustrates just how extensive this symbolic processing can be for cooking tasks.

4.6.2. Controlling Activity in Improvisation

As mentioned in Section 4.4, the control mechanism in any improvisational architecture must somehow control when the agent is biased toward using the most obvious ways of doing something or toward using deeper reasoning to find less obvious ways. It controls how long deliberation is allowed, and how extensive the search for alternative actions (to those directly recommended by routines) will be. It also controls how much physical effort an agent will put into some particular activity. In performing these tasks, the control mechanism must be extremely flexible. In making control decisions, it must take not only the external situation into account, but the agent's internal desires, expectations, and intentions. It must dynamically reconfigure itself in response to changes in any of these areas. Hayes-Roth [1993] notes that such a reconfigurable control methodology is crucial to any intelligent system designed for more than one small area of the spectrum of activity.

Clearly, this is a tall order for any system, but in moving toward a specific architecture for improvisation, there is one observation that can immediately be made. When performing everyday activities, *the agent's knowledge of the activities in which it is participating drives these choices.* That is, the known aspects of the activity *constrain* both the choices an agent can make in its activities and how attractive it is to put further intellectual and physical work into an activity. The use of constraints in everyday activities is directly observable in every example of recorded knowledge of everyday activities cited in Chapter 2: recipes provide constraints on how to use ingredients [Parker, 1886; Hadwen, 1937]; guides to environmentally-friendly living provide specific constraints on how we perform specific activities in order to reduce environmental impact [Cailliet et al., 1971]; and guides for recent immigrants provide constraints on how to go about everyday activities with respect to Canada's social and cultural norms [EmpImm, 1991]. This points toward viewing improvisation as a constraint-directed reasoning problem. Before moving to such a viewpoint, the following Section briefly introduces constraints as a knowledge representation mechanism.

4.7. Constraints

Inference in artificial intelligence usually thought of as deduction and search. Planning, in particular, has always been intimately associated with search (the search of a set of states for a plan to solve a given problem), because of a historical evolution that has been extensively detailed in Chapter 3. Indeed,

Simon [1983] does define search and deductive reasoning as two of the most important views of problem-solving. However, there is a third view of problem-solving categorized by Simon [1983], one that was largely unexplored in AI until the early 1980's, and one whose power is to a large degree only being realized today. This problem-solving methodology is known as *constraint satisfaction.*

Constraints are a knowledge representation mechanism based on restrictions: what is allowed and what is not allowed in particular situations. Formally, a constraint is a specification of a set of admissible assignments of values to variables. This can be by specifying domain constraints (e.g. $x \in \{1\ 3\ 5\ 9\}$) or by defining relationships between variables, such as $(x + 1) >= (2 * y)$ [Le Pape, 1992]. By defining constraints in this manner, any set of constraints c_i defines a subset of states in U.

Constraint satisfaction differs radically from search in its view of problem-solving. While search involves actively exploring a problem space, usually through some type of generate-and-test methodology, constraint satisfaction involves consistently enumerating values for a set of variables [Fox and Nadel, 1989]. As variables are given values, these values further constrain the choices for other variables. For example, if one were colouring a map according to the constraint that no two adjacent countries be the same colour, making a colour choice for one colour would constrain the choices for adjacent countries. As decisions were made for other countries, more and more constraints would be involved. The technique of extending or communicating constraints so that they define further relationships between variables is known *constraint propagation.*

When a set of constraints is used to characterize the solution of a given problem, that problem is said to be *overconstrained* when there is no solution that obeys the entire set of constraints. That is, $\nexists\ S_j \subset U \mid c_i \Rightarrow S_j,\ S_j \neq \{\varnothing\}$ (where $\Rightarrow$ is defined as "is described by"). In such cases, constraints may be selectively *relaxed:* the set c_i can be gradually weakened so that eventually a non-empty subset of U can be found that obeys the relationships the c_i define. Through propagation and relaxation, a set of values is found for the variables involved in the problem. This is usually a satisficing solution, because in many cases the set of constraints collectively eliminates all combinations, and constraints must be selectively relaxed so that a consistent set of variable values may be found. Because constraints are defined as relationships or connections between variables, a constraint satisfaction problem is often viewed as a graph or constraint network (Figure 4-8). The best analogy of a

constraint satisfaction problem to a physical structure is a child's "cat's cradle", where a long string is entwined between the fingers, forming a maze of strings (constraints), and the fingers may be moved about (selectively relaxing some constraints, and tightening others), changing the pattern of the network until a satisfactory structure results [Fox, 1991].

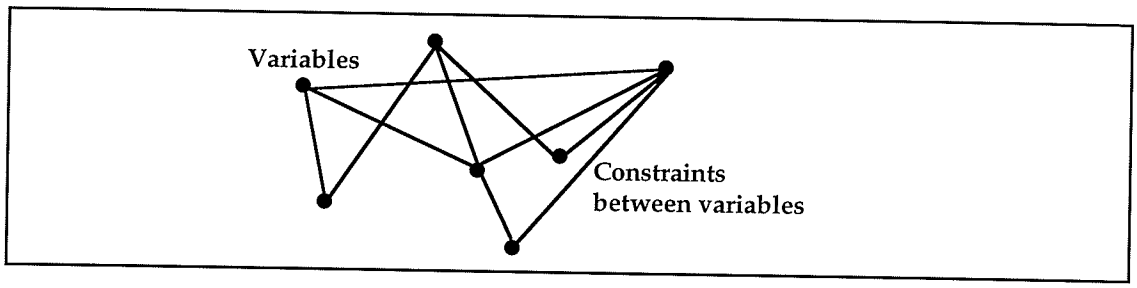

Figure 4-8. A constraint network.

Constraint satisfaction techniques have been applied to many types of problem-solving within artificial intelligence. While equivalent to search, in that the general goal of constraint satisfaction problem-solving is to search a universe of states to find a state that satisfies a given set of constraints [Rich, 1983], many problems become comparatively simple when viewed in this manner. In particular, problems that can be viewed as as *fit-to-specification* problems [Bond and Gasser, 1988] are naturally suited to constraint-directed reasoning. Such problems generally involve matching internal knowledge to a specific situation during problem-solving. Examples of this type of problem are map colouring problems (as described above), edge labelling in computational vision, the division of labour between multiple problem-solvers, and general design problems where a given design must be created under specific criteria. A survey of the many constraint satisfaction algorithms that can be applied to such problems may be found in [Kumar, 1992].

When constraint propagation and relaxation are employed in problems to which they are naturally suited, very elegant solutions often result. However, it is important to note that since these are essentially equivalent to search, they often have complexity problems themselves. When disjunctive constraints are allowed, the problem of determining whether a given set of constraints is consistent is NP-hard [Le Pape, 1991], and there may be large numbers of relaxations that could possibly be explored for any one constraint. Networks such as that shown in Figure 4-8 are visually eloquent, but become extremely large and complex in any significant problem. Many problems also

resist decomposition into such networks, as well: part of the problem may be naturally represented as constraints, while the rest may not [Fox and Nadel, 1989].

Because of this, another view of constraints is usually employed in artificial intelligence: that of *constraint-guided search*[54]. This perspective evolved from the recognized utility of constraints in constraint satisfaction problem-solving, and the desire to exploit the power of constraints in domains that are not well-suited to the constraint satisfaction approach. The general idea of the constraint-guided search perspective is that parts of a domain, encoded as constraints, can be used to describe the structure of the overall problem space, thereby reducing search. Fox expresses the power of constraints when applied in this way as follows:

> Domain knowledge, encapsulated in the form of constraints, can be used to develop a better understanding of the structure of the problem space which will lead to more efficient problem solvers whose architectures take advantage of known problem space structures, and whose focus of attention methods reduce the amount of search [Fox and Nadel, 1989, p. 77].

The most well known work in constraint-guided search is Fox's [1983] work on constraint-directed scheduling. This system (*ISIS*) performed job-shop scheduling, the process of ordering the flow of work through a factory consisting of a number of machines on which various operations can be performed in such a way as to maximize positive characteristics (e.g. throughput) and minimize negative characteristics (e.g. lateness). Viewing these characteristics and conditions in the world affecting the schedule as constraints, Fox modelled the scheduling process as a constraint-directed heuristic search for an adequate schedule By representing important aspects of the problem structure as constraints on the schedule, Fox's scheduler could focus on the most important factors affecting the schedule, vastly reducing the amount of search involved. For example, a bottleneck resource would be represented as a constraint, which would guide the search around that resource, thereby reducing the number of other paths that must be explored [Fox and Nadel, 1989]. Similarly, cost, time, and other factors were represented as constraints, further guiding the search for a satisficing schedule.

---

54 Together, constraint satisfaction and constraint-guided search techniques are referred to as *constraint-directed reasoning*.

The application of constraint-guided search to scheduling resulted in a system with scheduling abilities as good as any human job-shop scheduler. However, the system's real impact was its demonstration of the power of constraint relaxation: the ability to selectively weaken a combination of constraints out of a set, in order to find heuristic solutions to overconstrained problems. The system also demonstrated the flexibility of constraints as a knowledge representation mechanism in general. Constraints have also been used to allow a reactive component to scheduling: dynamically rescheduling when the environment around the scheduler changes. Such systems (e.g. *OPIS* [Ow et al., 1988], *SONIA* [Collinot and Le Pape, 1991]) have greatly increased the utility of automated scheduling systems.

Scheduling in systems such as ISIS can be viewed as one aspect of a larger general planning problem. Fox's implementation of job-shop scheduling benefits from a comparatively well-defined problem situation in that all of the steps required to complete a given order in a factory are defined, as are restrictions on the temporal orderings of the required steps. The scheduling task is one of providing a specific ordering on the various known steps of each job in a manner that satisfies as many constraints as possible. While this is itself an NP-hard problem [Fox, 1983], planning adds another dimension to the problem in that neither the exact operations required to solve the problem nor the temporal ordering of these operations are initially known[55]. Planning involves finding an appropriate, minimally-ordered sequence of actions to solve a given problem, while scheduling involves taking a plan and enforcing further ordering constraints to achieve robust and time-efficient execution of the plan [Fox and Kempf, 1985].

The most significant use of constraints in planning to date is Stefik's [1981a] *MOLGEN* planner, which used constraints in the construction of plans[56]; more specifically, in the planning of molecular genetics experiments. Constraints could be placed on objects manipulated by actions, allowing values that fit those constraints to be filled in later. For example, a step in a MOLGEN plan might be to filter a given bacterium out of a sample, and

---

55 From a scheduling point of view, this statement is accurate: planning in general is more difficult than scheduling. However, the scope of ISIS was much larger than that of most planning systems, in that it involved considering the ramifications of an action ordering on many other orders, and on the factory as a whole. In this sense, ISIS is in fact a much more complex system than most classical planners.

56 Again, an application of constraint-guided search.

might require that the bacteria that are to remain be resistant to a certain antibiotic. Maintaining this information in the form of a constraint rather than selecting a particular antibiotic right away helps prevent overcommitment and forcing backtracking later on (should the particular choice of antibiotic be unsuitable for some later step in the experiment). Actions are thus only partially instantiated (that is, the p_k for the action were left only partially described), providing a least-commitment approach to selecting objects within the plan. Each constraint could then be propagated backward through the existing portion of the plan, where it may have some effects on existing constraints. These constraint mechanisms were generalized and extended to domain-independent planning in *SIPE* [Wilkins, 1988], allowing the same mechanisms to be applied to virtually any planning domain.

While extremely successful, this is a very limited use of constraints in comparison to the representational abilities demonstrated by Fox [1983]. Of the many types of knowledge required in planning, MOLGEN applies constraint-directed reasoning to only one area: that of representing object relationships within plans. This limited use of constraints also does not apply to improvisation or everyday activities. Stefik's techniques were classical ones, and constraints were used strictly to control stepwise interactions in plan construction due to relationships between objects in the plan, rather than the much more complex interaction between an agent and its environment during activity.

Constraints have also been used to perform multi-agent planning, wherein multiple problem-solving agents can cooperate to solve more complex problems than any one agent can handle individually. This involves a great deal of reasoning over and above the level normally associated with planning: methods of decomposing tasks so that they may be shared, allocating tasks to appropriate agents, managing cooperation, and integrating results. Evans and Anderson [1989] have described an architecture that permits task sharing and negotiation between agents, and have extended that architecture to provide dynamic replanning capabilities [Evans and Anderson, 1990]. This architecture is also used as a basis for representing the autonomy of agents in multi-agent systems [Evans et al., 1992] using constraints.

This classical planning-oriented use of constraints has largely been the limit of the application of constraint-directed reasoning to intelligent agents. What is little appreciated in planning is that plans themselves are simply collections of constraints, whether a planner views them as such or not. A

classical plan is a collection of actions constrained in a particular order. The reasons for each action and the order in which they appear (what Sacerdoti [1988] describes as *teleological* knowledge) have simply been compiled down to a single ordering constraint. A simple example is shown in Figure 4-9.

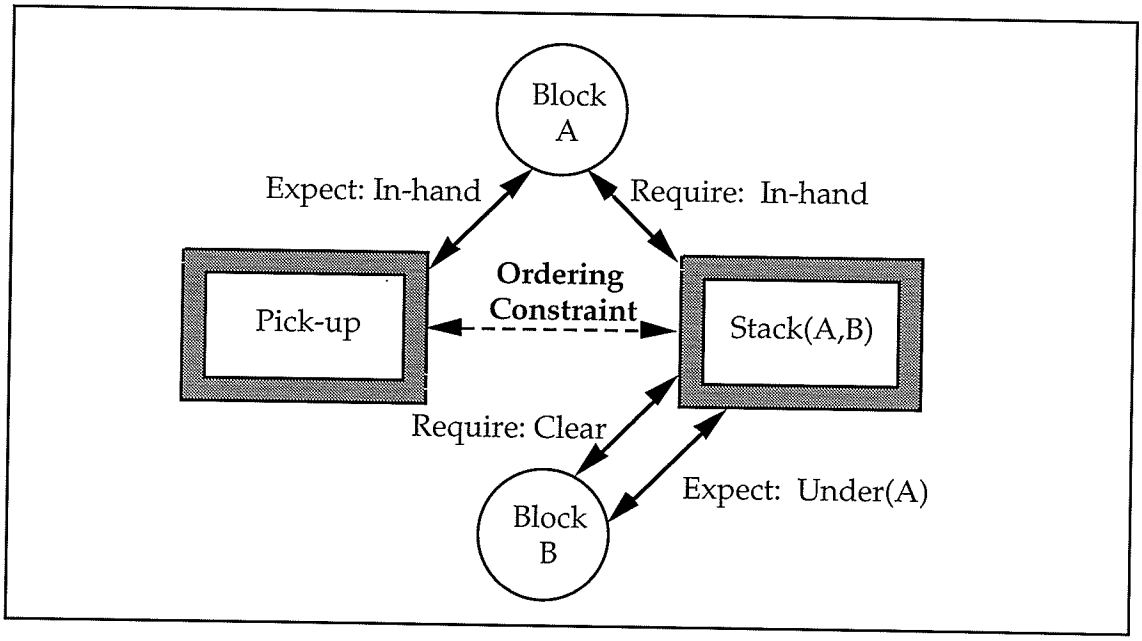

Figure 4-9. Reasoning behind action ordering condensed into a simple ordering constraint.

Here, two blocks world[57] actions are connected by some specific requirements: the pick-up action allows the block to be placed in the agent's hand, a state that is required to be able to stack the block on top of another. There is an ordering constraint imposed on these actions, which the agent uses to perform one action before the other. However, this ordering constraint is the sole knowledge the agent possesses; the planner producing the plan has reasoned about the sequence of actions necessary to stack the block, recognized that the pick-up action provides for the requirements of the stack action, and generated the constraint from this knowledge. Having condensed this

---

57 The *blocks world* is a common example of a toy domain in AI problems, and is used almost universally for examining techniques in planning and testing planners. It involves a series of blocks that may be moved and stacked using a mechanical arm.

knowledge, the underlying reasons for the constraint are discarded[58]. In this example, we are throwing away only a few pieces of knowledge; however, in any significant example, the reasons behind any ordering constraint will be very complex (involving the behaviour of the objects involved in action, other motives possessed by the agent, the behaviour of other agents, etc.). The reduction of all of this knowledge to a simple constraint in a plan results in an an inflexible, overconstrained structure. Similarly, a universal plan is a set of constraints describing the actions that are suitable in a particular situation, and is similarly overconstrained.

In fact, constraints have been vastly underutilized in planning: while complex planning concepts have continually pointed to the use of constraints, they have rarely been exploited directly. For example, in case-based planning, goals are often referred to as being *blocked* by a given situation in the environment [Hammond, 1989b; Schank and Abelson, 1977]. Much of Hammond's recent work in case-based planning [Hammond et al., 1990] involves reasoning about blocked goals and the situations in which they may be satisfied. Introducing constraint-directed reasoning would allow the planner to reason directly about the overconstrained situation causing a goal to be blocked, and relaxing any number of constraints in order to achieve that goal.

The various uses of constraints in planning and scheduling described above are all instances of a few general categories. Fox and Nadel [1989] argue that constraints may be used to perform many general functions, applicable to planning as well as other areas of AI. They can be used for *pruning:* eliminating unlikely prospects during search. They can also be used for *rating:* estimating the cost of violating or following constraints, allowing determination of the feasibility of following a given path. Constraints may also serve as *integrators*, indicating dependencies between parts of a problem, or indicating the restrictions generated by choices made so far in the solution

---

58 The exception to this is in modern interleaving planners, which keep track of the requirements and provisions for each action, allowing later actions to be discarded if the provisions they meet are available for other reasons. This allows the planner to take advantage (in a limited manner) of serendipitious events. Even in such planners, maintenance of requirement/provision information is the limit of the use of background knowledge. Such information is maintained by sophisticated interleaving planners such as *IPEM* [Ambros-Ingerson, 1987], but was also used in more basic systems such as the *triangle tables* (described briefly in Chapter 3) used in *STRIPS* [Fikes et al., 1972]. Maintenance of more complex knowledge is far beyond the capabilities of such planners.

process. They can also be used to *focus* problem-solving efforts on the most tightly or loosely constrained parts of a problem.

In addition to planning and scheduling, each of these general functions applies directly to the requirements of improvisation. Indeed, I would argue that improvisation more clearly demands each of these functions than does either planning or scheduling. The limited amount of search possible during improvisation demands exceptional pruning and rating abilities from an improvising agent; likewise, the distributed nature of improvisation demands that restrictions generated by current activity be integrated with those of intended action in the future, and vice versa. The extreme range of activities and the wide variety of knowledge of those activities available to the agent also demand an exceptional ability to focus; the dynamic nature of the worlds that improvising agents inhabit demand the ability to alter that focus rapidly. All of these needs point directly to the use of constraints as a knowledge representation and control mechanism for improvisation.

4.8. Summary

This chapter has described improvisation, an approach to everyday activities based on the timely integration of compiled knowledge and the background knowledge from which those compilations are derived. A precise definition of improvisation has been provided, and the general requirements of the computational processes an improvising agents must embody, and the knowledge representation problems that an improvising agent must deal with have been outlined. I have also characterized improvisation as an intelligent control problem, and have illustrated the promise that constraint-directed reasoning shows toward dealing effectively with this problem. Finally, I have briefly reviewed the use of constraint-directed reasoning in artificial intelligence, with particular regard to its potential for application in everyday activities. Throughout this Chapter, I have endeavoured to discuss these aspects of improvisation at a general level, without regard to any particular architecture for improvisation.

The following Chapter describes a constraint-directed architecture for an improvising agent, answering the requirements outlined in Chapters 2 and 3 and embodying the principles of improvisation and constraint-directed reasoning described in this Chapter. This architecture shows constraints to be a useful representation mechanism for the routines required by improvisation, in that constraints have the flexibility to represent both the highly compiled routine knowledge and the connections to the deeper

knowledge that lies behind that core. The architecture also demonstrates that constraint-guided search is an useful control mechanism for applying these routines, in that constraints can represent the knowledge necessary to apply a routine and control decision-making under resource bounds, and can be relaxed to allow the agent to perform in a satisficing manner.

CHAPTER 5

A CONSTRAINT-DIRECTED APPROACH TO IMPROVISATION

Constraint (n.) The state of being checked, restricted, or compelled to avoid or perform some action.

– *Webster's Ninth New Collegiate Dictionary*

We are working at an arrangement in form, of the myriad disparate details of housework, family routine, and social life. It is a kind of intricate game of cat's-cradle we manipulate on our fingers, with invisible threads...

– Anne Morrow Lindbergh, *Gift from the Sea*

5.0. Introduction

Improvisation in everyday activities has been characterized as applying an agent's knowledge of activity under resource bounds to choose and carry out actions that are appropriate in terms of the agent's immediate goals and other activities the agent intends to carry out in the future. This Chapter describes an architecture for an intelligent agent capable of accomplishing everyday activities by means of improvisation. This architecture is based on two founding premises: first, that constraints are an ideal mechanism for representing the many regularities in everyday activities; and second, that constraint-guided search can be used to exploit these regularities, focus the agent's intellectual resources toward useful behaviour, and control the

deliberation of an agent during improvisation. The description of this architecture is divided into three parts. First, the nature of constraints and constraint-directed reasoning in improvisation is examined. This serves as a basis for the organization of knowledge within this architecture. Finally, I describe the mechanisms by which knowledge structures are applied to produce improvised behaviour.

5.1. Constraints, Improvisation, and Everyday Activities

In Section 4.7, numerous uses of constraints in decision-making were detailed, both in general and as applied to scheduling and planning. In improvisation, constraints can serve several important functions. Constraints can be used to focus resources to the most appropriate activities, for example, by restricting the agent from squandering resources on unimportant activities. Constraints can also be dictated by the agent on the environment, serving a regulatory function on future behaviour. For example, an agent may expect certain results from an action or some event in the environment, and use this expectation as a restriction on its further behaviour. Most significantly, however, constraints can be used to represent restrictions dictated by the environment itself, serving as *guides* toward or away from particular courses of activity.

As mentioned briefly in Section 4.7, any classical plan is a collection of actions with ordering constraints, and these actions and ordering constraints summarize or condense the large collection of background knowledge from which the plan is derived. We have also seen in Chapter 2 that this collection of background knowledge is rich and varied. If one were to examine all the individual pieces of knowledge necessary to apply a single recipe, for example, one would soon be overwhelmed. However, a linear list is an extremely artificial means of viewing this knowledge: there is in fact much structure available that may not immediately be noticed. It is this structure that constraints are ideally suited to represent.

A view of the structuring of everyday activities (from Figure 2-2) in terms of constraints is shown in Figure 5-1. As described in Chapter 2, the natural structure of everyday activities makes the process of choosing appropriate alternatives during the course of activity obvious: specific pieces of knowledge either eliminate unfruitful alternatives or guide the agent toward appropriate alternatives. So it is with constraints: by representing both general knowledge of activity and knowledge of the specific domain in the form of constraints, specific constraints can eliminate choices or serve to

guide the agent toward particular choices. Both approaches serve the purpose of turning a normally deep or wide decision structure into the shallow or narrow structures that characterize everyday activities.

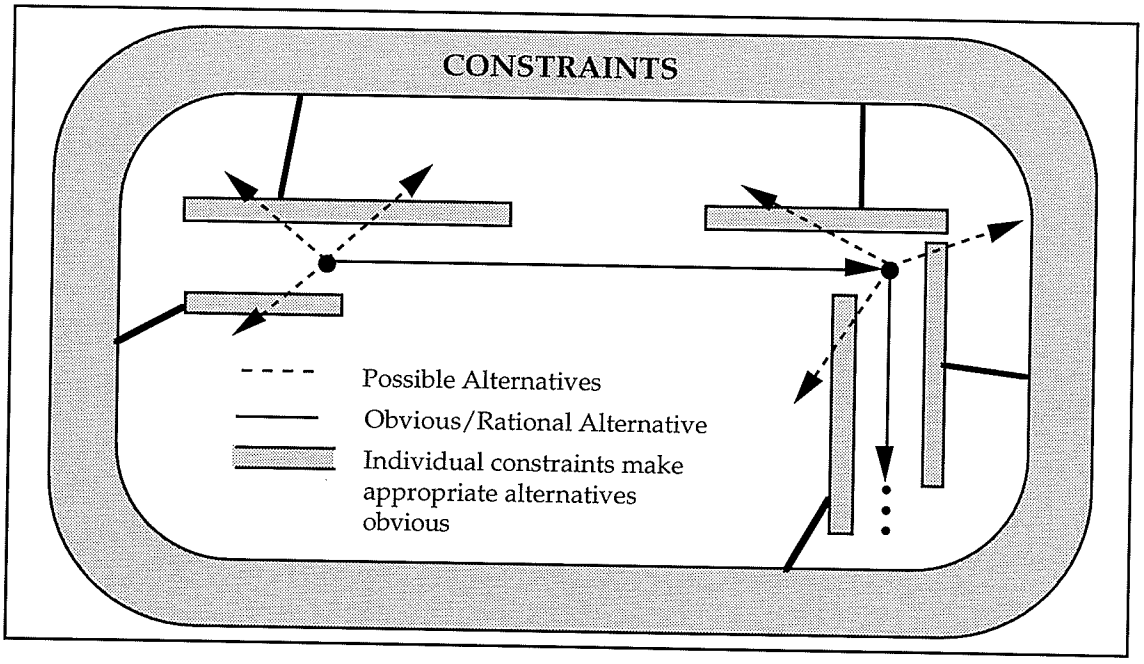

Figure 5-1. Constraints represent the structure of everyday activities.

In fact, there are three ways that constraints can guide activity, not all of which are obvious from Figure 5-1. First, a constraint may restrict the agent by eliminating one or more implausible or inappropriate alternatives, allowing consideration of a smaller number of more reliably valid alternatives[59]. A constraint may also compel the agent toward a course of activity by lending support to alternatives that obey the constraint. This could also be viewed as constraining alternatives that detract from some course of activity. Finally, a constraint may have a number of obvious relaxations associated with it, automatically bringing about additional potential actions when a constraint cannot be obeyed. The latter function is important to constraint-directed reasoning: it allows constraints to function as generators within a search space, as well as evaluators of particular states [Fox, 1983].

---

59 Minsky [1986] calls a piece of knowledge knowledge applied in such a manner a *censor*.

The fact that large parts of a domain are representable using constraints has been well-demonstrated, most notably by Fox [1983]. However, the use of constraints proposed here takes a very different perspective on the use of constraints than that of Fox. In Fox's view of job-shop scheduling, constraints represent demands on the scheduler, and the constraint-directed scheduling process became a matter of deciding "How can a schedule be chosen which satisfies as many constraints as possible? And if a constraint cannot be satisfied, which relaxation of a constraint can be satisfied?" [Fox, 1983, p. 58]. Improvisation is inherently different than this. *When improvising, an agent need not attempt to satisfy as many constraints as possible: rather, the constraints represent the natural structure of the activity, and guide the agent to rational choices* (as illustrated in Figure 5-1). In this view, constraints no longer *demand* satisfaction, but serve as visible indicators of how the agent should carry out its activities. Rather than asking how we can satisfy as many constraints as possible, we ask how we can use the constraints inherent in everyday activities to guide the agent's use of limited intellectual resources while accomplishing its desires in a satisficing manner. Improvising agents make the best decisions for action they can given the constraints on their behaviour imposed by the environment around them and their intentions for future behaviour.

Constraints have previously been shown to be very powerful when used as guides. In design, for example, Baykan and Fox [1987] have shown that the running time of a space planning program is most greatly affected by the strength of the constraints available in the design problem: more so in fact than the size of the design problem itself. Interpreted in the domain of everyday activities, the stronger the guidelines, the more effective the agent. Baykan and Fox [1987] also illustrate the promise shown by the use of constraints to identify opportunistic decisions.

Thus far, I have described the general role that constraints are expected to play in improvisation. This discussion, however, has said little about exactly how those constraints are to be applied. In order to develop an architecture for improvisation based on constraints, we need to decide what types of information are best represented as constraints, how these constraints are to be organized, and the types of processing these constraints require. In short, we need to further examine the nature of constraints in everyday activities.

5.1.1. Structure of Constraints in Everyday Activities

As mentioned in the previous Section, constraints can serve many purposes in everyday activity, the most significant of which are their use as guides, describing the structure of everyday activities and allowing the agent to focus its efforts appropriately with regard to that structure. Within this role, constraints have many specific purposes:

- Constraints can serve as *inhibitors of activity*: they can inhibit the agent from working with certain objects, performing certain actions, or participating in certain types of activities. This inhibition may be in the form of a complete debarral, a quantitative rating on some scale, or a qualitative measurement. For example, a social setting imposes many constraints barring or inhibiting an agent from performing actions that might be perfectly acceptable were the agent alone.

- Constraints can also serve to *compel* the agent to perform certain actions, or to expend extra effort toward a given activity. To use the same example, in a social setting an agent is constrained (sometimes quite severely) to perform particular actions at particular times (such as actions or communications to be made when greeting or parting).

- Constraints can *represent interactions* between actions or activities, or between an action or activity and some part of the world around the agent. For example, the agent may, upon picking up and using a spoon during cooking, be constrained to clean the spoon and put it in its proper storage space.

- Constraints can serve to *control* the amount of physical effort spent on an activity, the amount of cognitive effort spent in making a decision, or the amount of consideration given to other constraints. For example, the agent may restrict the amount of time spent deliberating when a certain type of action is involved, or the amount of time spent deliberating toward a certain activity.

- Constraints can also serve to *represent the agent's expectations of the future*. When an agent performs an action or witnesses an event, for example, the expectations the agent makes of the action or event is used to base future activity. These expectations can be represented as restrictions on a future world: restrictions whose violation may lead to significant changes to the agent's intended activities.

- Constraints can serve to *limit the extent of improvisation*, controlling the amount of effort spent on finding some improvised alternative during the course of an activity. As stated in Chapter 4, improvisation can range from fairly mundane variations on a particular activity to more extensive jumps of imagination. This can result in great variation both in the amount of effort put toward an activity and the appropriateness of the eventual solution. Restrictions on these aspects can be used to control how far along these dimensions improvisation can proceed.

- Constraints can serve to *modify an agent's routines* (Chapter 4) based on current conditions within the environment. A routine activity such as grocery shopping can be greatly modified by the addition of a constraint that it must be done quickly, for example.

- Constraints can serve to *limit the recollection* of agents and the amount of information that is examined during the course of activity. As discussed previously, an agent performing everyday activities encounters many objects in its surroundings that may serve as indications for further activity. These objects may be perceived as the agent goes about its activities, or may be recalled from memory as part of an ongoing course of activity. Constraints can serve to inhibit the amount of mental exploration that an agent performs when it encounters new objects: the extent to which it explores its memory for connections with current or future activities.

One of the most powerful features of constraints as a representation mechanism is that in addition to the wide range of roles that constraints can play, all constraints have a similar underlying structure, allowing very different aspects of a problem to be treated in a similar manner computationally [Fox, 1983; Evans and Anderson, 1989]. In everyday activities, all constraints share the following attributes:

- *Activation conditions.* Each constraint has a set of criteria under which the constraint is applicable. A constraint may only be applicable in the presence of a certain object, for example, or may only be useful when some event has occurred.

- *Focus.* A constraint is not only applicable under certain conditions, it is also applicable to a certain entity. A specific constraint may apply to a given object for example, or a given potential action. The collection of

objects or attributes to which a given constraint applies will be referred to as the *focus* of the constraint.

- *Dependencies.* In addition to activation conditions, a constraint may have specific pieces of knowledge on which it is explicitly dependent. These pieces of knowledge may alter the behaviour of the constraint or possibly invalidate it completely.

- *Lifetime.* Each constraint has a specific lifetime during which it is active. This lifetime may be an objective period of time or may be a set of conditions describing when the constraint should become inapplicable once it is active.

- *Utility.* Each constraint also will have some measure of its utility or significance. This utility will vary by context, and may be expressed quantitative or qualitatively.

- *Effort.* In addition to measuring the utility of a given constraint, an resource-bounded agent must be concerned with the amount of effort needed to satisfy or obey the constraint.

- *Relaxations.* When a constraint cannot be satisfied, it may specify general or particular relaxations that may be used in lieu of the constraint. Each relaxation will also have associated measures of utility and effort.

While all of the above attributes have aspects unique to the domain of improvisation, most have been used previously in applications of constraint-directed reasoning (e.g. [Fox, 1983; Evans and Anderson, 1989; Evans et al., 1992]). One unique aspect of many of the above attributes is their strongly context-dependent nature with respect to improvisation and everyday activities. Because attributes such as utility and effort are context dependent (e.g. the amount of effort for an activity such as going to the store to get milk varies greatly by the time of day), these measures may not be immediately obvious during deliberation, and indeed, initial guesses may be totally dependent on many particulars of the situation in which the agent finds itself. However, because of resource bounds such measures may not be completely explored. Control mechanisms must exist to allow satisficing, resource-bounded estimates of such attributes as utility and effort, calculated based on the importance of the activity. This moves beyond previous applications of constraint-directed reasoning such as [Fox, 1983; Evans et al., 1992] that used fixed estimates for such attributes.

5.1.2. Types of Constraints in Everyday Activities

Recognizing the roles of constraints and the the similarities shared by all constraints in everyday activities is the first step in applying constraint-directed reasoning to this area. To proceed further however, we must focus on the differences between constraints based on how they are employed. We must identify the specific *types* of constraints necessary to express the structure of everyday activities, in order to define the knowledge representation and manipulation mechanisms required. One of the major contributions of Fox's [1983] work in job-shop scheduling was in demonstrating that the wide variety of knowledge encompassed in this domain was representable using only a few different types of constraints. That is, that there was a great deal of underlying similarity in this knowledge that could be taken advantage of by a constraint-directed representation. This is also a characteristic of everyday activities and improvisation: despite the extremely wide variety of knowledge involved (as demonstrated in Chapter 2), only a relatively few types of constraints are necessary.

Because of the similarities between scheduling and improvisation, in that both involve the resource-bounded determination of ordered actions based on domain-specific criteria, many of the categories of constraints described by Fox [1983; Smith et al., 1986] are equally applicable to improvisation in everyday activities. Many other types of constraints that are particularly emphasized in everyday activities can be considered sub-types of Fox's categories. A similar relationship between scheduling and multi-agent planning was demonstrated by Evans and Anderson [1991; Evans et al., 1992]. This relationship also exists between multi-agent planning and improvisation: having characterized improvisation as an intelligent control problem (Section 4.5.1), much resemblance exists between the knowledge required for improvisation and the control knowledge necessary for coordinating multiple agents. This similarity is also reflected in the types of constraints necessary for improvisation in everyday activities.

All knowledge necessary to support improvisation in everyday activities can be described by one of seven different types of constraints. These constraint types, as well as important subtypes, are shown in Figure 5-2. In order to further emphasize the connection of these constraint types to everyday activities, examples are drawn (where possible) from the guidebook discussed in Section 2.2, describing some of the knowledge surrounding many everyday activities in Canada [EmpImm, 1991].

In any domain in which everyday activity takes place, *physical* constraints, representing physical restrictions on objects in the environment, are necessary. A door may only allow objects of a certain width to pass through it, for example. Such constraints also extend to the agent itself as a physical entity (e.g. a limitation will exist on the weight of an object the agent can lift safely). Physical constraints may also define physical relationships between objects, as opposed to characteristics of single or typical objects. Geographical or containment relationships are an example of this, as are less obvious relationships such as hierarchical or "part-of" relationships. Exclusion relationships through physical constraints are also possible: for example, two actions may be mutually exclusive or may require an explicit temporal ordering due to physical interactions.

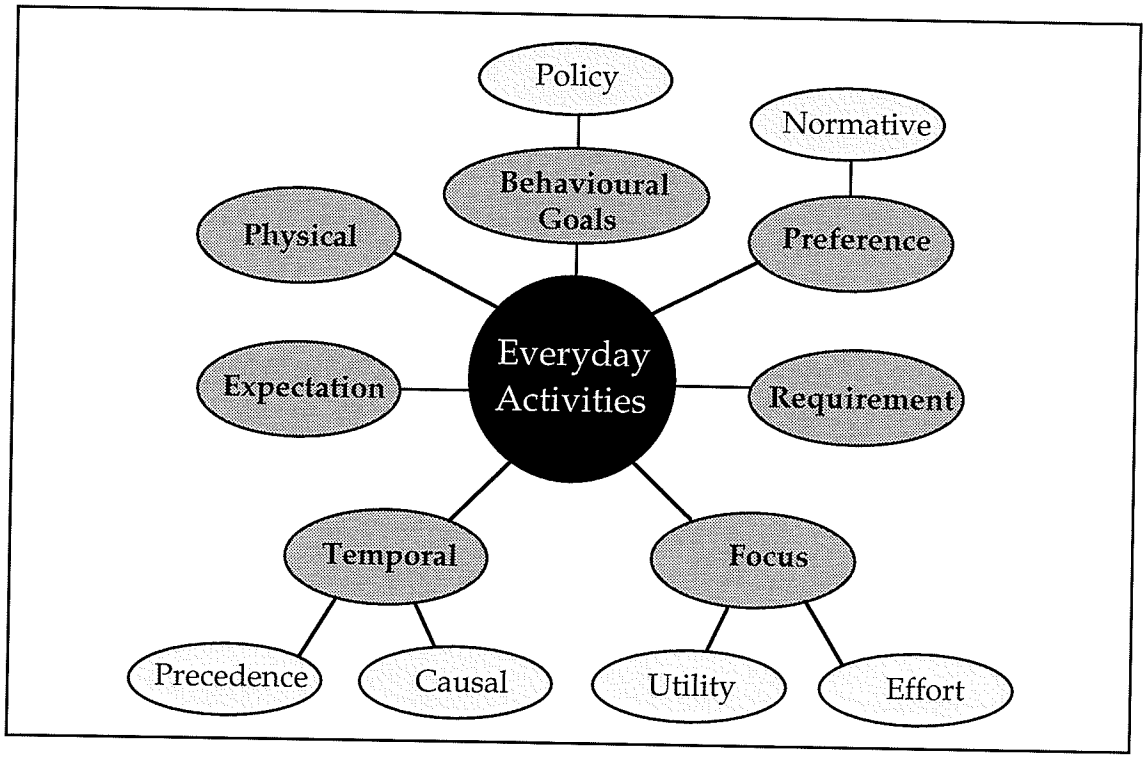

Figure 5-2. Categories of constraints for improvisation in everyday activities.

Requirement constraints are constraints between the agent's intentions and the present world around the around the agent. They define the resources and internal states required for some resource to be applicable to a given activity. Such resources may be physical objects (e.g. tools), metaphysical concepts (e.g. time), states of particular objects in the world (e.g. water for tea

must be boiling) or internal states within the agent itself (do *X* only if not doing anything else). Requirement constraints can also be applied in a social sense. Such constraints require the agent to commit to certain actions in particular situations: for example, being on time for a job interview [EmpImm, 1991].

Expectation constraints bear some resemblance to requirements, in that they are constraints between the agent and the world around it. However, an expectation constraint defines a relationship between the world in which the agent finds itself and a world in the future, as opposed to the present. For example, the agent may expect a given object to be at a particular location as a result of an action. An action may also be expected to take a certain amount of time or energy to complete. Expectations bear some analogy to the results produced by actions in earlier models of activity, such as the add/delete lists of STRIPS [Fikes and Nilsson, 1971]. However, expectations entail much more than simple state changes, since they represent all constraints on what the agent *expects* will come of an action. Such expectations may include much more abstract concepts than simple facts about the environment: the amount of time results are expected to take, or the expected cost in terms of some resource, for example. The agent can then use these expectations as a basis for reasoning about further activity, as a gauge to whether problems may be occurring in current actions (e.g. it's taking too long for some result to appear), and as a gauge for what can be considered as unexpected results. Like requirement constraints, expectations also have a social aspect. Any action involving other agents may have expectations of the behaviour of those agents, such as expecting a friendly response from a greeting, for example [EmpImm, 1991].

Preference constraints are another important category of constraint in activity. A preference constraint restricts one entity to be preferred over another in some manner. In activity, many types of entities may be involved in a preference constraint: one tool may be preferred over another, one action over another, one plan over another, or even one method of comparing tools or actions over another. Fox [1983] noted that preference constraints are in one sense an abstraction of the attributes of the entities involved: a preference relationship is maintained rather than the many individual reasons for the preference. Just as plans substitute for the detailed reasoning that goes into their construction (Section 4.1), preference constraints substitute for the detailed reasoning that lies behind them. For this reason, preference constraints are utilized heavily in reactive reasoning. Universal planning, for example, can be viewed as using a strict set of preference constraints: a

given situation constrains the agent considering only one specific action for the agent to carry out.

In improvisation, preference constraints are used where sufficient information or time does not exist to evaluate all the knowledge behind them, as described in Chapter 4. However, while preferences are useful as caches of abstracted information, it is important that they not be relied upon too heavily. Abstracting out the factors involved in a preference makes for easy comparison, but it does so by removing precisely the information required to understand *why* one entity is better than another [Minsky, 1986][60]. Too strong a reliance on preferences creates the same difficulties as too strong a reliance on plans.

A useful subgroup of preference constraints are *normative* constraints, which define what is considered "normal" in a particular situation. For example, it may be considered normal to answer the telephone when it rings; it is normal to put dishes in a sink when they are dirty; and it is normal to consider a potential purchase carefully when it costs a lot of money. Normative constraints may also be specific to the agent itself, rather than activity in general: it may be normal to have coffee in the morning, for example, or normal to eat dinner around 5:00 PM. These represent general preferences for behaviour in that they constrain the agent to an appropriate response in a given situation: normal responses may be preferred over abnormal ones, simply because they have (presumably) been successful in such a situation many times before.

Temporal constraints are mainly used to describe restrictions concerning time as a resource or the temporal ordering of entities. An agent may wish to set bounds on the amount of time devoted to a given activity, or to deliberating about a given decision. A constraint ordering two actions (e.g. Figure 4-9) can also be considered a temporal constraint. Most temporal constraints are obviously identifiable as such; however, there are two significant subgroups that are more subtle in nature. First, temporal constraints can also define *precedence* in a relationship. When defined on actions, for example, a precedence constraint may state that some other action must be performed

---

60 In everyday activities, many preferences exhibit this phenomena. There are many cases in routine situations where preferential methods for going about activities can be explained in terms of background knowledge, but also many others where preferences are so strong and background information so rarely used that the origin of the preference is lost.

before or after it [Smith et al., 1986]. The same type of restriction is often seen in everyday activities: for example, one must pay for goods before leaving a store [EmpImm, 1991]. The former action is not a requirement for the latter: rather, this restriction says that a payment action must take precedence over an exiting action. Temporal constraints can also identify *causal* relationships between actions and states in the future. Causal knowledge can be represented as restrictions that a given action places upon the objects it affects. Thus, if milk is left unrefrigerated, it will spoil. This can be represented as a constraint on any action placing the milk in an unrefrigerated state.

Another important category of constraints is that of *behavioural goals*. These are very similar in nature to the organizational goals used in job-shop scheduling [Fox, 1983]. In Fox's model, organizational constraints were used to reflect the high-level goals of the organization for which scheduling was being performed. These constraints represented restrictions on the cost of items, delay times, the amount of work in progress, etc. These were viewed as approximations of a simple profit constraint, and allowed measurement of scheduling decisions in terms of the "costs" they would incur and the "profits" that would be realized at the organizational level [Fox, 1983]. An individual agent has similar goals concerning its behaviour. At any particular time, it may wish to minimize cost, or be particularly careful at some activity (thus using more resources). The agent may wish to hurry a particular everyday activity, or to take its time. All of these are examples of behavioural goals. Such high-level constraints provide a context within which other constraints are interpreted.

One particularly useful subgroup of behavioural goals are *policy* constraints. Policy constraints restrict the agent's behaviour to general policies it is used to following. For example, a policy constraint may restrict the methods the agent can use to achieve what it wants (e.g. don't hurt anyone) or the types of behaviour that should be performed (e.g. if you use a tool, put it away afterward). Policy constraints may also restrict specific aspects of certain behaviours (e.g. always look for all possible interactions before action X is carried out; limit time constraints whenever an action of type Y is being considered).

The concept of a policy has been used before in research on planning. McDermott [1978] included entities known as policies in an early theory allowing reasoning about individual actions outside of classical plans. Policies, in this theory, were secondary tasks that allowed alterations to the manner in which primary tasks were carried out. For example, a primary task might be to buy some object, and a secondary policy such as "always have

money" that the agent was constantly carrying out would cause the agent to modify its actions to replace the money it spent. Parallels between McDermott's use of policies and those used here can be drawn. However, policies are viewed here not as modifiers of a task but as *general principles of activity*. Policies are not constraints that drive activity when violated: they are not goals in disguise. Rather, like other types of constraints, they are restrictions that influence the agent's decision-making. The variety of constraints included as policies in this model is also much wider than those proposed by McDermott.

Policies are applied in a more wide-ranging manner by Hammond et al. [1990], who extend McDermott's basic use of policies by making the assumption that the *only* method that causes activity to arise is the interactions of policies with the environment. Again, this use of policies differs greatly from that used here, mainly because policies do not drive behaviour through violation within this model.[61]

The final category of constraint useful in performing everyday activities is that of *focus* constraints. These constraints allow the agent to focus its efforts appropriately when dealing with many potential activities by placing restrictions on the resources used during improvisation and deliberation. In job-shop scheduling, this focus was provided through knowledge associated with each constraint regarding relevance (when a constraint should be considered), degree of satisfaction (the utility of alternatives), and importance (influence of the constraint on the scheduling process) [Smith and Ow, 1986]. This knowledge is also directly applicable to constraint-directed reasoning in everyday activities. The relevance of a constraint is easily maintained, due to the nature of knowledge in everyday activities, and will be explained in Sections 5.2 and 5.3. Applying the latter two types of knowledge is more complex, however, and requires the use of two types of focus constraints. *Utility* constraints set limits the usefulness of alternatives to be considered during improvisation: they can be used to restrict consideration to only very closely compatible alternatives, or to allow consideration of anything that might possibly satisfy the agent's desires, no matter how unusual. *Effort* constraints set limits on the amount of effort and time spend deliberating about alternatives, and thus control how responsive the agent is to the

---

61 This position will be explained in Section 5.1.4. The model of Hammond et al. is described in Section 8.4.

environment around it. The actual application of focus constraints to control deliberation will be explained extensively in Section 5.3.4.

In addition to having a wider range of types of constraints in comparison to job-shop scheduling, any everyday activity will also have a much wider variety of individual constraints. From Chapter 2, this should be fairly obvious: any everyday activity can involve a very large range of knowledge. When compared to the variety of constraints necessary to model a job-shop, as was done in Fox's [1983] ISIS system, it can be readily seen how simplistic the job-shop scheduling domain really is relative to everyday activities. Fox [1983] shows that exactly twenty-one constraints, each falling into possibly multiple categories, are collectively required to define a job-shop scheduling domain. Compare this to the range of constraints necessary to give even a partial high-level description of a simple routine for an everyday activity: Figure 5-3 illustrates just a few of the very wide variety of constraints that are involved in background knowledge for making tea.

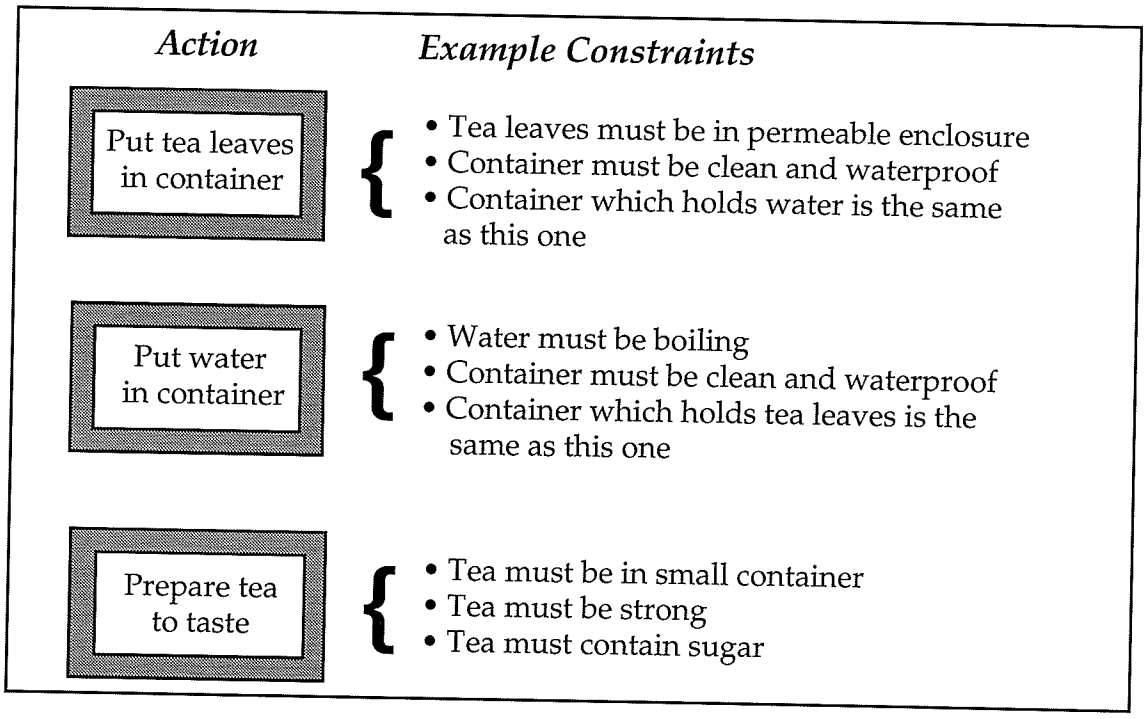

Figure 5-3. Variety of constraints in a simple routine.

There are many constraints omitted in this Figure, not only from individual actions (e.g. all constraints involving putting tea leaves into a container are not shown) but more importantly, between actions and between this activity

and others. While some constraints between actions are illustrated (e.g. *the container that holds water must be the same as this one*), many are omitted. There are many constraints that appear at higher levels than that shown in the Figure: for example, how tea-making can relate to other activities. In addition to the categories of constraints described here, there is an obvious organization to constraints with regard to the entities in the environment they are defined on and between. This organization is described in the next Section.

5.1.3. Organization of Constraints in Everyday Activities

The previous Section has shown that constraints are appropriate for representing a wide range of knowledge pertaining to activity. However, the categories presented previously say very little about how these types of constraints apply to individual objects or how they interact to produce the behaviour observed in everyday activities. A preference constraint, for example, serves a wide variety of purposes, from preferring one tool over another to selecting a default action without reasoning about the criteria behind that default. Categories of constraints serve as the raw materials for the design of an improvising agent. In order to develop such an agent, we need to examine the fundamental structuring of these constraints in everyday activities: the entities that participate in the relationships the constraints define, and the manner in which those entities relate to one another. By further organizing constraints by the entities on which they are defined and the purpose for which they are employed, the structuring of knowledge necessary for an improvising agent becomes clear.

In Fox's job-shop scheduling domain, this structuring was relatively straightforward (Figure 5-4), due to the relatively limited number and variety of entities involved in the reasoning process. Constraints could be between and among particular orders, machines, and operations, resulting in a two dimensional structure. More complicated domains, such as multi-agent planning, require a hierarchical structure in order to deal with the added complexity of allowing higher level groupings of objects to have constraining effects [Evans and Anderson, 1991]. For example, in the case of multi-agent planning, membership in a given social group provides a role for an individual agent, thus constraining that agent's behaviour. In everyday activities, actions can be part of larger behaviours, defining relationships between one another.

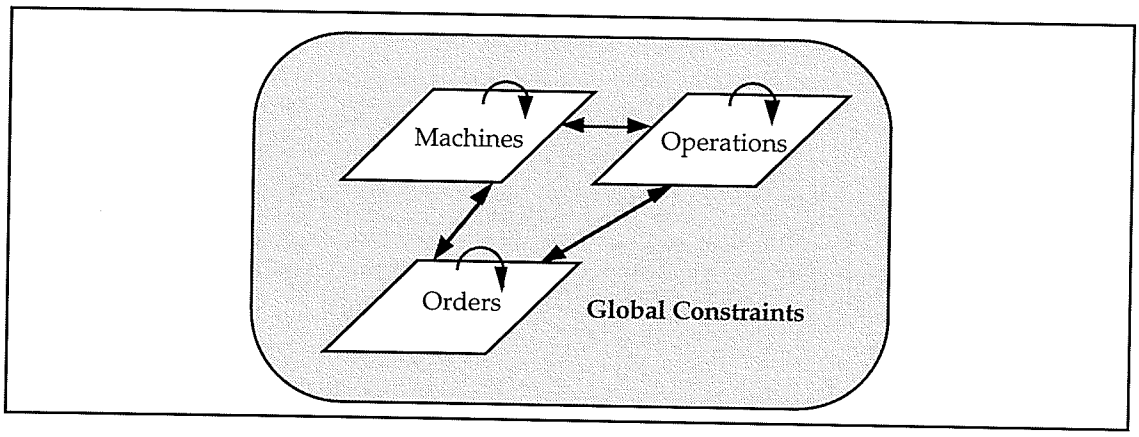

Figure 5-4. Organization of Constraints in the job-shop scheduling domain of ISIS [Evans and Anderson, 1991].

Like multi-agent planning, improvisation in everyday activities involves a far richer domain than job-shop scheduling: the number of type of objects involved and the ways in which they relate to one another are vastly larger in everyday activities. Accomplishing everyday activities also involves reasoning at multiple levels, reflecting the subsumptive nature of these activities (as described in Chapter 2). The organization of constraints for improvisation in everyday activities therefore involves a naturally hierarchical organization, illustrated in Figure 5-5. Many of the categories of constraints described in the previous Section are applicable to multiple levels of this hierarchy (e.g. requirements, expectation); others are more obviously associated with one level than others (e.g. behavioural goals).

The hierarchy depicted in Figure 5-5 divides constraints into four levels based on the entities upon which they are defined. These levels will be discussed shortly. Beforehand however, there are several points worth noting about the nature of this hierarchy of constraints. First, the hierarchy is not a strict one: while each level builds upon the knowledge contained in the levels below it (e.g. actions are clearly part of action sequences, and thus the constraints contained in actions are applicable to a sequence of actions), lower levels can also affect upper levels outside of this hierarchical relationship. The hierarchy shown in Figure 5-5 also abstracts both knowledge and control. Each level represents knowledge not representable at lower levels. For example, in a sequence of actions forming a routine for making tea, knowledge about putting a tea bag in a cup and putting hot water in a cup would be maintained at the action level, while the fact that the cup must be

physically the same object in both steps for the correct outcome would be stored at a higher level.

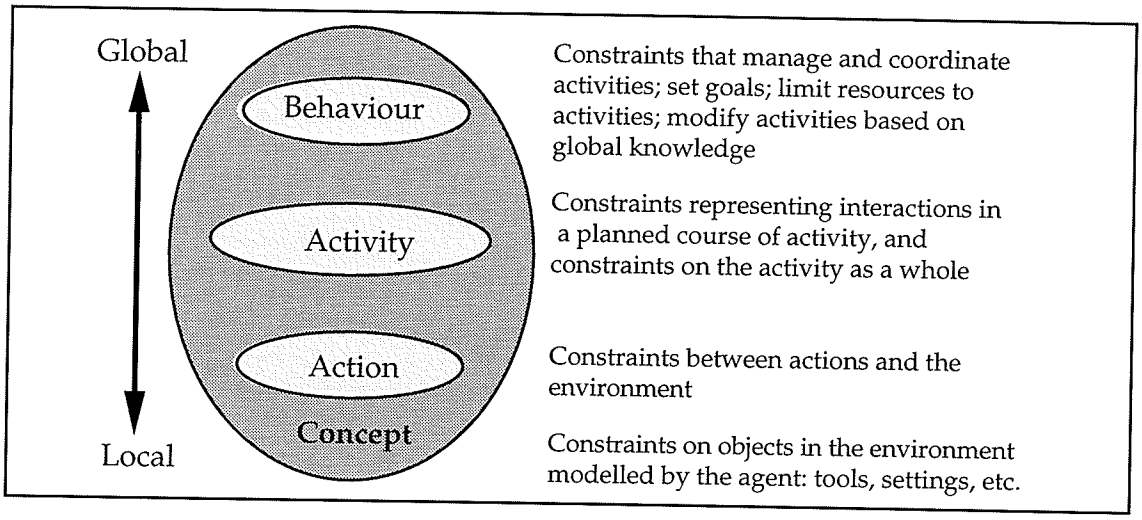

Figure 5-5. Organization of constraints in improvisation.

The level that one most associates with constraints in planning systems is the *action* level. Constraints at this level place restrictions between a particular action in the knowledge base of the agent and the environment in which the agent is performing. A specific action will have given requirements, for example, constraining the action to certain states of the external world. Any action will also involve expectations of the changes it makes to the world around the agent. Constraints may also be defined on the interaction between a particular action and the agent itself (as a physical part of the world). A given action may affect the agent's internal motivations, for example, or interact with some higher-level concept the agent is considering. The latter is a situation where a lower level of the hierarchy affects constraints at a higher level. For example, an action such as answering the telephone may lead to the activation of a whole series of constraints for social behaviour, as well as for commitment to talking on the phone as opposed to immediately hanging up and doing something else.

The *activity level* records constraints on and within plans[62]. These constraints may be amalgamations of lower level constraints (e.g. the expected

---

62 I use the term *plan* here in the sense of plans-as-guides, as described in Section 4.1. However, this could also apply to constraint-directed classical plans.

time for a group of actions to be carried out would be an amalgamation of the expected times of its member components). Other constraints at this level allow reasoning about interactions between the various lower-level actions that form the sequence (e.g. as mentioned above, the fact that in a two-action sequence of putting a tea bag in a cup and pouring hot water over it, the cup must be the same in both actions to achieve the intended result). Still other constraints record information about the sequence itself: its requirements and purposes, for example, as well as the objects it uses and the expectations it generates. In the same manner as the action level, constraints at the action-sequence level can also affect higher levels.

Above the activity level is the *behaviour level*. The behaviour level consists of constraints that restrict various aspects of the courses of actions the agent may consider following, and represent interactions at the activity level. Such constraints apply to activity as a whole, and represent high-level guides toward the kinds of actions (e.g. what they should contribute toward or avoid, what kind of effort they should expend) the agent should be performing. A behaviour-level constraint may be focused on a particular course of activity (e.g. expend only a very small amount of effort toward this activity) but are independent and exist separately from knowledge of that activity. Behavioural goals fall at this level: policies such as *don't run out of resources*, for example, define policies for carrying out intentions. This constraint would serve to remind the agent to consider working to acquire more tea bags if it has very little or if its intended activities would involve using a significant quantity of tea[63]. Constraints at this level can also serve to perform specific modifications to a course of activity. If time is short, for example, a *hurry* constraint may be placed on a given course of action, thereby altering many aspects of how that activity is performed. Constraints at this level may also serve to limit processing at lower levels, to restrict how other constraints act and the amount and variety of information that is involved in decision-making, as well as the level of resources devoted to particular activities or individual actions.

---

63 Such reminders serve as the basis for the architecture of Hammond et al. [1990]. As mentioned in the previous Section, however, these reminders act very differently in this architecture, and represent only one of many ways in which constraints are employed. The differences between this work and that of [Hammond et al., 1990] will be explored in Section 8.4.

In Chapter 2, rationality in everyday activities was characterized as balancing local and global knowledge (both in the form of current and future activities and in the form of local and global preferences). The behavioural level directly supports this rationality, by providing a global view of the activities in which the agent is engaging and the goals that it is following.

Each of these levels provides context to the levels below it: action-sequence constraints, for example, provide some governance over constraints at the action level. Likewise, behavioural constraints provide a context for interpreting lower-level knowledge. For example, a state of caution may be invoked through a behavioural constraint, that would in turn affect how the agent performed many activities (it might, for instance, cause it to check that all the necessary ingredients were available before making tea). Or, the telephone may begin to ring during an activity like making tea, invoking a behavioural constraint increasing the importance of answering the phone. This high-level behavioural constraint provides a context for evaluating other actions the agent may be contemplating. However, while the behaviour level is the most obviously important in terms of control for decision-making, there is another level that is far more pervasive in improvisation. This is the *concept* level, and involves constraints on all of the objects in the environment, both physical and abstract, of which the agent has conceptual knowledge. In any particular everyday activity, the concept level will involve by far the largest number of constraints, as it defines a constraint-directed conceptual model of environment the agent inhabits. Each physical object or class of domain objects known to the agent has some concept associated in the agent's memory[64], and each of these concepts has constraints (of various categories) associated with it. An agent's concept of milk, for example, may have a causal constraint indicating that if it is left out, it will spoil. Preference constraints may indicate preferred means of performing some action using the object, and requirement and expectation constraints may illustrate particular conditions associated with the object and its use in a given action. Abstract concepts are also associated with this level. For example, while the setting in which an activity takes place is often thought of as a physical one, settings can also be abstract, consisting only of a collection of knowledge constraining activity (a social setting, for example).

---

64 Chapter 4 illustrates the significance of memory to improvising agents.

The concept level is placed not at any one particular point in the hierarchy, but surrounding the other levels. The concept level occupies this position because it affects all other levels equally, and is thus both locally and globally constraining. Actions, for example, involve constraints on all the physical objects they will affect, as do action sequences. The use of some particular object can also activate global constraints at the behaviour level. Think of the constraints on behaviour that are instantly invoked when picking up a gun, for example, as opposed to picking up some other object. While all the other levels also affect one another outside of a strict hierarchical relationship, only the concept level is so universally applicable. In fact, the levels described so far can all be viewed as providing an interpretation on the constraints at the concept level: interpreting the constraints provided by the agent's concepts of the world around it, in light of the actions, action sequences, and high-level behavioural goals the agent is considering.

There is one more thing that can be said about the organization of constraints in everyday activities based on the description of improvisation and everyday activities provided here. Chapter 2 has demonstrated the memory-intensive nature of everyday activities, while Chapter 4 described how memory interacts with other aspects of improvisation. Because of this memory-intensive nature, the problem of determining constraint relevance (one of the three major properties of constraints defined by [Smith et al, 1986], as described in the previous Section) can be dealt with fairly easily. Since every aspect of the environment, from physical objects to actions, to abstract concepts such as settings, will have some conceptual framework in the memory of an improvising agent, constraints can be organized as attachments to the mental concepts they are defined upon. Thus the constraint knowledge concerning tea, for example, can by physically attached in memory to the agent's concept of tea. Such constraints can then automatically become relevant whenever that concept is referred to[65]. This process will be explained further in Sections 5.2 and 5.3.

---

65 This relates to Norman's [1988] work on design for ease of everyday activities: devices in the world should remind us of the constraints associated with them, and clearly indicate how the device is to be used. For example, the mechanisms controlling doors should somehow visually indicate whether the door should be pushed or pulled to open it [Norman, 1988].

5.1.4. Constraint Processing in Everyday Activities

There remains one additional area that must be explored before embarking on a detailed description of the architecture that is the main focus of this Chapter, and that is the manner in which constraints are used to guide activity. I have argued that constraints can serve as guides to activity, and have shown that the organization of constraints required for improvisation is very different from that of scheduling and planning. The manner in which constraints are applied in improvisation is equally distinct from these areas.

Constraint-directed reasoning in improvisation can be divided into two important aspects. Some constraints contribute toward the agent's behaviour through their violation. For example, when the agent picks up a cup from the counter, it generates an expectation constraint, expecting the cup to be in the agent's hand a short time after the action is begun. Such a constraint must be actively monitored via perception, and if violated, will (presumably) drive activity to repair the violation. This manner of viewing constraints is crucial to both planning and scheduling applications: constraints are imposed, and after such imposition, must be actively monitored. Such a view is equally crucial to some aspects of everyday activities. Like the example above, once an expectation is made it must usually be monitored: it is an assumption on which further behaviour is based. If the assumption proves incorrect, the behaviour relying on it may have to be altered.

The majority of constraints in everyday activities, however, do not act in this manner. Most constraints in everyday activities are used as guides on an agent's behaviour, and drive behaviour not through their violation, *but through the effect the constraints themselves have on deliberation*[66]. Such constraints are used to influence the agent in some fashion, either by making some alternative attractive (e.g. a constraint such as *keep milk cold* may support an alternative for action such as replacing milk in the refrigerator when the agent brings it home from the store) or by eliminating unattractive alternatives.

---

66 By this I mean not only the physical process of deliberation, that is selecting one action out of many, but also in the many processes that support deliberation. For example, constraints on how much information is to be retrieved from memory, how perception is to be biased, how expectations of previous actions drive deliberation, how extensive a search for alternatives should be before deliberation begins (or during deliberation), etc.

Viewing the majority of constraints in improvisation in this manner is absolutely necessary. Suppose, instead of the above view, we chose to see the constraint on keeping milk cold as a put *perishable items in the fridge* constraint, which is violated the moment the agent gets home and the milk is not in the refrigerator. This would clearly invoke the same course of action; presumably to put the milk in the refrigerator. However, the manner in which such activity is invoked is entirely inappropriate for everyday activities. In this case, the agent now has a violated constraint as soon as it enters the house (in fact, the constraint is violated as soon as it picks the milk out of the cooler in the store, and remains so until the milk is once again refrigerated). The agent must then decide whether to work to satisfy the violated constraint, or to pursue other activities and ignore the violation. For a single constraint this is not a problem, and indeed this view seems intuitive. For any significant number of constraints, however, this is an impossible view, because the number of constraints that can be actively monitored in such a manner is necessarily small. Consider for a moment the hundreds of potential constraints associated with any everyday activity. Were all constraints to be treated in this manner, the agent would have to spend virtually all its time simply monitoring the world for things that could possibly affect its current set of constraints. It would have no time for performing any real work.

In addition to being impossible to implement for everyday activities, the traditional view of constraints used in AI systems is also highly unnatural philosophically to everyday activities. When performing some everyday activity such as making tea, we *do not* spend all of our time evaluating every possible minor problem before proceeding. Continually checking and evaluating every aspect of every potential action and every object known to the agent results in anxious, frantic, paranoid behaviour, because of the artificial[67] view of constraints as obstacles. *In everyday activities, each constraint represents some part of the overall structure of the activity, including its relationship to the environment around the agent and other activities the agent wishes to perform. Collectively, these constraints influence the agent's decision-making, just as natural features of the landscape influence one to travel in a given direction. Viewing all constraints as driving activity through their violation is not only impractical,*

---

67 For the purpose of everyday activities.

it directly contradicts the inherent intellectual simplicity of everyday activities.

In fact, the *only* constraints that produce activity through their violation are expectation constraints as exemplified above. All other types of constraints influence activity directly through knowledge contained in the constraints themselves. Most of the types of constraints detailed in Section 5.1.1 are fairly obviously of this variety. The only type that does not immediately fit into this framework is that of requirements. It may at first glance make sense to see requirements as constraints that drive activity through violation. For example, say the agent is making tea: one of the obvious requirements during the course of the activity is boiling water. Assume for the moment that not having boiling water available is viewed as a violation of the requirement constraint of having boiling water. This causes a problem, because there are a number of very different scenarios that cannot be differentiated. For example, say the agent is in the process of boiling water, and something goes wrong (e.g. the stove breaks). The same requirement constraint would be violated. The same constraint would also be violated if the agent intended to have tea sometime in the future, and something came along jeopardizing its ability to obtain boiling water (e.g. we know the stove is broken). The same result, a violated requirement, occurs each time - yet these are very different situations, to which the agent will respond entirely differently. Requirement constraints, like all others save expectations, are meant to be used as guides, influencing agent to work toward getting hot water as opposed to doing something else. They do not drive activity through their violation.

This is not to say that *no* activity arises due to the violation of constraints: only that the sole category of constraints that operates in such a manner are expectation constraints. In the second of the three scenarios presented above, for example, violated expectation constraints drive activity: the broken stove violates the expectation of having boiling water that is associated with the action that put the kettle on to boil. More importantly, it is also not to say that violations cannot occur to other types of constraints. Constraints other than expectations *are* violated, in all three cases. There are requirement constraints associated with needing boiling water for future actions to occur, in all these cases, and when boiling water is unavailable, they *are* violated. These violations simply do not direct future activity in the manner of violated expectation constraints. While they may trigger further constraints that many in turn influence activities, we do not monitor these constraints, nor care in particular about their violation. In the case of this example, suddenly not being able to get boiling water does not necessarily invalidate

the whole activity; rather, the requirement of needing boiling water will directly guide the agent toward activity appropriate to this requirement. Other constraints may support this, or may influence the agent toward performing other actions.

Having demonstrated the power of constraints in representing various aspects of everyday activities, and given an outline of their organization and the perspective from which they are to be viewed, we can now begin to examine how constraint-directed reasoning can be put into practice in improvisation during the course of everyday activities.

5.2. Knowledge Structuring for Improvisation

Throughout the previous Chapter, I have emphasized that an appropriate representation of knowledge is the key to improvisation. The architecture described in this Chapter relies on several forms of knowledge representation, some seen almost universally in AI systems, some completely novel. This Section describes each in detail, giving examples of each structure's use in everyday activities.

5.2.1. Intentions

The bulk of an agent's knowledge of everyday activities within this architecture is organized into structures known as *intentions*. These knowledge structures contain both the routine knowledge that forms the bulk of everyday activities, as well as deeper knowledge that supports more extensive improvisation, as detailed in the previous Chapter. Before describing these knowledge structures in more detail, I feel the need to justify the use of the term *intention*, and more importantly the use of yet another form of knowledge structure in planning. It is commonplace in new theories of planning to see both completely redefined terminology, and "new" methods of knowledge representation, often bearing only superficial changes to generally accepted mechanisms.

In this case, a new mechanism for knowledge structuring is justified. As illustrated in the previous Chapter, improvisation is a spectrum that demands both the representation of routine aspects of an everyday activity (i.e., the obvious way of performing some everyday activity in a particular situation), as well as the deeper knowledge that forms the basis for that routine. This knowledge of the objects, the techniques, and the physics involved in the everyday activity allows the agent to follow its routine in a

wide variety of situations. The fact that this is beyond the scope of classical and universal planning and requires a knowledge structure supporting both types of knowledge has already been been demonstrated.

The choice of the term *intention* is a more subtle one. In addition to avoiding the term *plan* (and its historical connotations), the term *intention* is well-suited to describe the two levels of knowledge required for improvisation. Intention is most commonly thought of in terms of anticipated action, in the sense that one may intend to do something at some point in the future [Bratman, 1987]. The concept of intent, however, is much broader than this. One can perform some action with the intention of accomplishing some explicit purpose [Bratman, 1987], for example. This view seems to equate the concept of intent with the goal or desire behind a particular plan or action.

There is another view of intention, however, one in which intent is not characterized merely in terms of goals, but rather, stresses the active role of intentions in guiding agents toward those goals. Indeed, Miller et al. [1960] describe an intention as the uncompleted parts of a (classically-oriented) plan whose execution has begun, making intentions and plans essentially indistinguishable. This is a fairly extreme view. However, Boden [1973, p. 24] argues that the view of intentions as guides to activity arises naturally from the concept of intentions as goals:

> References to intention in everyday life...are regarded as explanatory of action because the intention is conceived of as somehow bringing about, causing, or directing the action intended. The intention to buy bread, for instance, somehow results in the act of purchasing a loaf. This functional feature of intention is emphasized if one thinks of an intention as a schema controlling the procedures to be executed in behaviour, as a plan of action that can be realized in actual operations carried out by the agent. The essential aspect of any given intention is thus some procedural schema or action-plan. The crucial question at this stage is to ask what must be the nature of an action-plan, in order that it may be able to carry out the controlling function commonly ascribed to intentions.

The concept of an intention as a collection of knowledge guiding activity is also expressed by Austin [1970, p. 283]:

> The most subtle of our notions is that of intention. As I go through life, doing, as we suppose, one thing after another, I in general always have an idea - some idea, my idea, or picture, or notion, or conception - of what I'm up to, what I'm engaged in, what I'm about, or in general 'what I'm doing'. I don't 'know what I'm doing' as a result of looking to see or otherwise conducting observations: only in rare and perturbing cases do I discover what I've done or come to realize what I am or have been doing in this way...I must be supposed to have *as it were* a plan, an operation-order or something of the kind on which I'm acting, which I am seeing to put into

> effect, carry out in action: only of course nothing necessarily or, usually, even faintly, so full-blooded as a plan proper. When we draw attention to this aspect of action, we use the words connected with intention.

This sense of the term *intention* precisely describes the purpose of these knowledge structures: to guide activity without at any time dictating what the agent must do. From this point onward, when the term *intention* is used, it will be in the sense of a particular structuring of knowledge for improvisation, containing all the knowledge necessary for the performance of a particular everyday activity.

Intentions serve two functions. First, they provide a set of flexible guidelines for a course of activity to accomplish some desire. They represent the agent's previous experience with the activity. They also express regularities in the problem structure (aspects the agent has seen time and again), and constrain search by making responses that have worked in the past obvious. The constraining action of these routine aspects can also be relaxed, allowing the agent to access the deeper constraints out of which the routine arises. The flexibility inherent in these guidelines gives the agent the ability to apply the intention over a wide range of situations. That is, the knowledge in the intention is suitable not only for the precise routine the agent has performed many times before, but also in a wide range of variations on the activity, conforming to the range of improvisation shown in Figure 4-2.

In addition to guiding the agent's individual actions in order to contribute to a particular goal or desire, intentions also provide an influence over the agent's attention at a higher level. Intentions constrain the agent's choices for action to those that are compatible toward the activity in question [Bratman, 1987]. If the agent were intending to go to a restaurant, for example, this intention would constrain future choices for activity that preserved the possibility of going to a restaurant, and eliminate those that threatened it. This allows the agent to keep track of many activities, including those set in the future, and act without necessarily invalidating any of them. These two functions are essentially the same, viewed from two different levels.

Given the requirements of everyday activities, intentions must support two competing aspects. First, given the simplicity and ease of deliberation of everyday activities, they must allow fast access to routine aspects of an activity (i.e. the obvious way must be obtained quickly). This would seem at first glance to advocate the storage of specific routines. However, this is unfeasible, because of the extensive symbolic manipulation required to apply such specific routines in varying situations. If we were to use the routine

shown in Figure 4-7, for example, and the first action was for some reason invalid, we could not immediately move on to others: symbolic plan repairs would have to be performed. Even if a routine could be adapted to a different situation, performing the required tailoring is known to be very difficult [Hammond, 1989a], making the exclusive representation of routines a poor choice from an improvisational point of view. On the other hand, representing only high-level versions of any activity, and then calculating specific ways of performing the activity suitable to particular situations has similar problems. This was the paradox described in Section 4.6.1.

There are two points to note when examining a detailed sequence of actions such the tea-making actions shown in Figure 4-7 that indicate how this paradox can be resolved. First, many of the actions in the routine are not directly part of what we would consider making tea: getting milk out of the refrigerator, for example. These parts of the tea-making activity are really separate everyday activities on their own: that is, the agent has a routine for obtaining milk when it needs it in the kitchen, and applies this routine in much the same fashion as the overall tea routine. The need to obtain milk is a core part of the tea intention, but the details of doing so are a separate activity.

We can also immediately note that the sequence of actions shown in Figure 4-7 is not at all what we would come up with if we were asked to describe how we make tea: its far too detailed. These details must somehow be easily arrived at, yet not be the sole means of reasoning. That is, the agent must have access to the specific means of accomplishing parts of an intention, methods that the agent may or may not consider "automatically" when carrying out its intention. For example, filling the kettle, turning the stove on, and waiting for water to boil is simply a specific method for boiling water, one to which the agent might automatically default if no information obviously affecting that method were available. This is how routines work: we don't actively explore how to do each step that we've performed thousands of times before: we simply do it automatically when it seems all right to do so.

Intentions resolve the paradox described earlier through a distributed representation based on the two observations made above. Just as everyday activities are hierarchical, so are intentions: the activity of making tea encompasses the activity of putting milk in the brewed tea, which in turn encompasses the activity of obtaining the milk from the refrigerator. Part of the intention of making tea is dealt with by adopting an intention to add milk to the tea, which in turn is partly dealt with by adopting an intention to go to

the refrigerator and get it. Each intention provides a context (a set of constraints) within which to interpret the intentions below it.

The resulting structure of intentions is shown in Figure 5-6. An intention consists of a very brief, abstract description of the activity it encompasses, together with the relevant background information (concepts embodying objects, settings, tools, actions, etc., and the constraints attached to them) that forms the foundation of the activity. The centre portion of Figure 5-6 shows a general description of how one goes about making tea. This description is necessarily skeletal in nature for the purposes of this Figure: some preference and requirement constraints are indicated, giving some order to the actions indicated, but many other constraints are not shown (e.g. the fact that the brewing container may be the same as the drinking container).

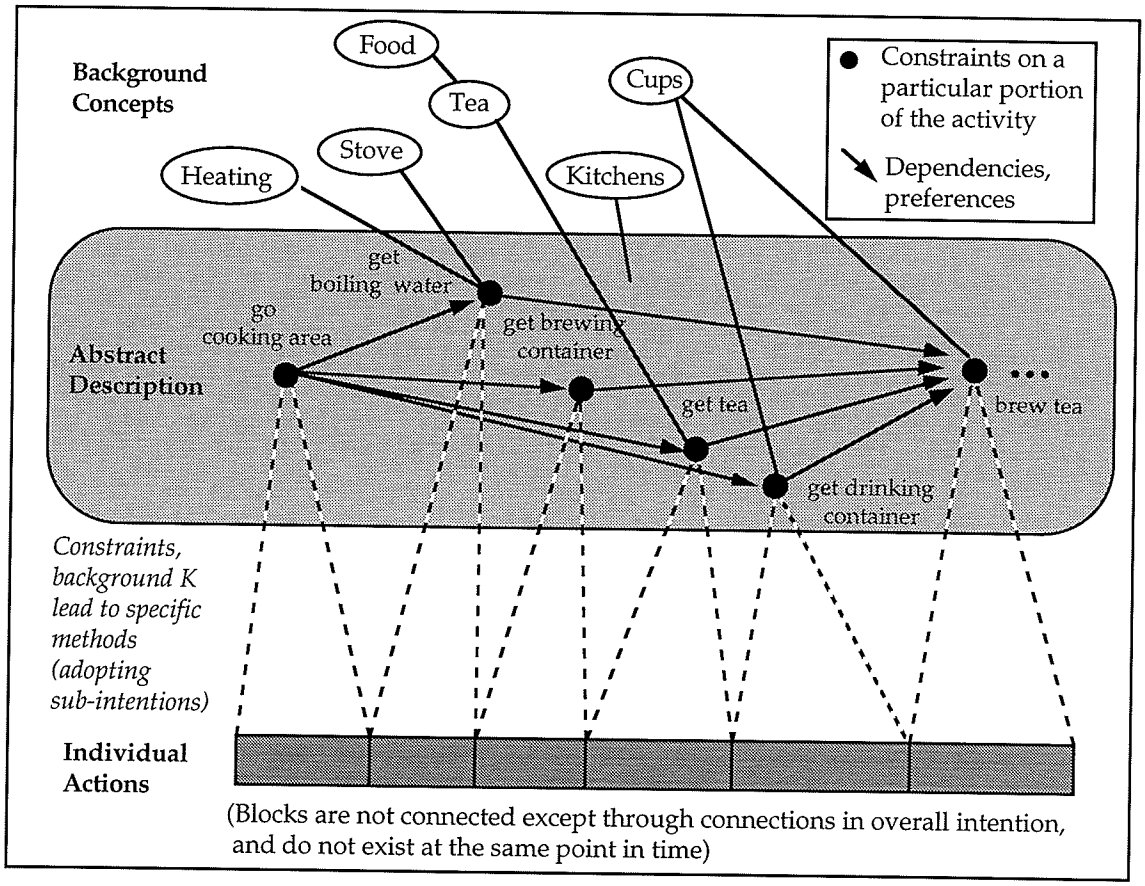

Figure 5-6. Organization of an intention, also illustrating the activity of making tea.

Many of the constraints in an intention will be provided through links to background knowledge. As seen in the Figure, background concepts can apply to many activities within a single intention, and may also contribute constraints at higher levels (the general kitchen setting, for example, applies to the entire activity, and will thus contribute constraints at all the levels described in Section 5.1.2. This background knowledge is organized in the form of multiple hierarchies, consisting of common object and process hierarchies (e.g. a *cup* is a type of *utensil*) as well as groupings in other ways convenient to activity (e.g *milk* is a type of food *food*, but is also *spoilable*). This structuring is described in Section 5.2.4. Constraints are attached to these objects, guiding activity or limiting the consideration of other constraints. Intentions also contain constraints on specific parts of the activity (e.g. *get-brewing-container* may have a constraint that defaults to a certain specific type of object, a certain methodology for the activity, or a certain place to begin looking for such an object), and constraints on the activity as a whole (e.g. limits on the amount of effort to be expended toward making tea). Background knowledge is highly interconnected, and one concept will normally be part of many intentions. Any other intention that normally takes place in a kitchen will share a connection to the *kitchen* setting shown in Figure 5-6, for example. Because of the interconnected nature of background knowledge, the boundaries between intentions are indistinct, as illustrated in Figure 5-7. This indistinct nature is intuitive in the nature of everyday activities: as described in Chapter 2, the knowledge used in performing one activity carries over to many others, and the activities themselves blend together into an ongoing pattern of behaviour.

Turning to the abstract description in Figure 5-6, it is especially important to note that the preference constraints that allow a default ordering do not *dictate* the ordering of the various activities within the intention: they advocate or restrict a particular choice of action, just as other constraints will advocate or restrict alternatives. In the tea intention of Figure 5-6, for example, there is a series of operations that may be performed before the *brew-tea* step. A preference constraint may have the agent begin by boiling water (again, this simply represents a habit) rather than looking for a cup, or any of the other actions. If there were any reason other than preference to take one action over another, constraints imposed by the activity or the environment would encompass these reasons, and would recommend alternatives. Intentions such as that shown in Figure 5-6 may also be affected extensively by constraints external to the intention: for example, during the course of making tea in an unfamiliar setting, the agent may have trouble finding a kettle to boil water. This may trigger a behavioural constraint to be

more careful for the rest of the activity, invoking further actions associated with the intention that might not be considered otherwise, such as checking the existence of all necessary ingredients before proceeding further.

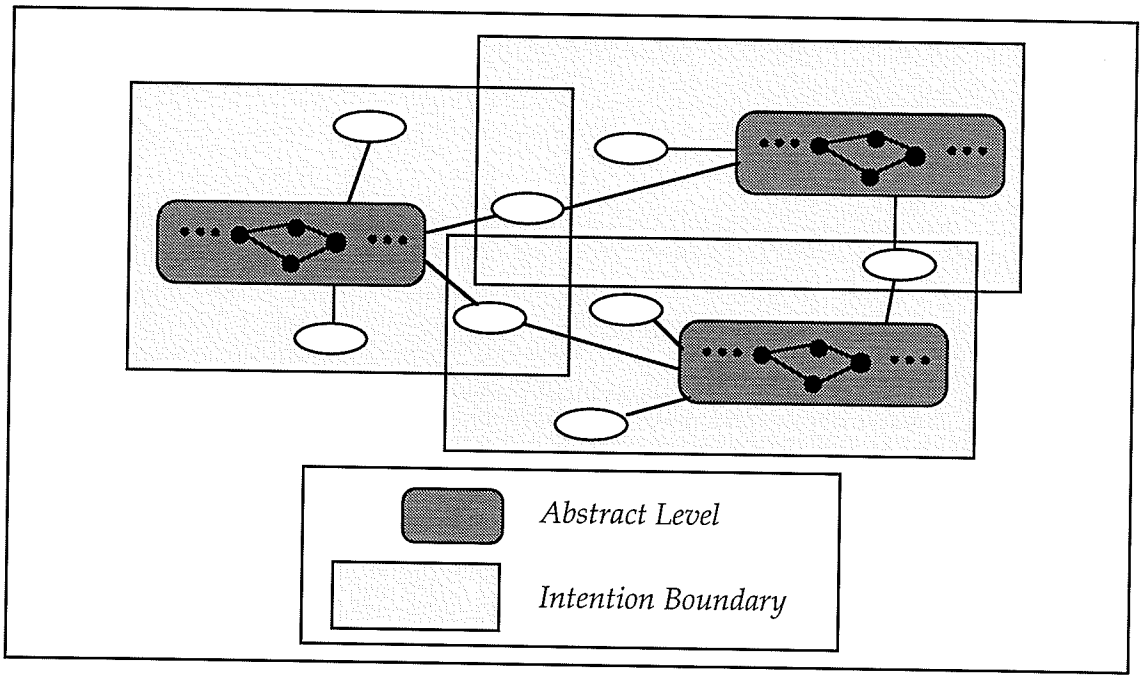

Figure 5-7. Intention boundaries are indistinct due to shared concepts.

As activities within an intention are committed to, they may in turn cause additional intentions to be adopted. Deciding to obtain boiling water, for example, would lead to considering methods of doing so, and would ultimately involve an intention to accomplish this, completely contained within the higher-order tea intention. The higher-order intention may have one or more preference (usually normative) constraints that cause the agent to default to some usual or preferential method of accomplishing this intention. For example, the agent might automatically use a stovetop kettle unless there was some obvious reason not to do so. Eventually, this will become simple enough that atomic actions are involved, which can simply be carried out (the blocks at the bottom of Figure 5-6).

The preference constraints that allow the agent to quickly adopt new intentions by default to satisfy some ongoing intention are used frequently: they are compiled knowledge that the agent can take advantage of in familiar situations, as described in the previous Chapter. The background knowledge performs a reciprocal function, allowing the agent to reason beyond such

surface preferences when the situation warrants (again, as described in Chapter 4). This supports the fast access to lower-level routine actions that improvisation demands: in a familiar setting, the agent can simply rely on the preference constraints to guide it to more detailed intentions with little or no cognitive effort. Consider once again the tea-making example. If the agent *always* uses a stovetop kettle to boil water, the first step in the routine may be to turn the stove on: *not* consciously asking *now, how shall I boil water* and then deciding to do it in a particular manner. From the point of view of intentions, the agent arrives at the point in the intention where it must boil water, and without any consideration, automatically adopts an intention to use a stovetop kettle. In a less-familiar situation, this may not occur. The preference constraints may be contingent upon some conditions that must be supported by the environment (e.g. stove available), or more likely, in inappropriate situations other constraints will provide stronger alternatives for activity, and the preference will not even be considered (e.g. knowing the stove is broken may set up series of constraints that avoid doing anything that might involve the stove in the first place).

1. Find a kettle and a teacup.
2. Fill the kettle with water, put it on the stove, turn the stove on.
3. Get a tea bag, put it in the teacup.
4. When the water is boiling, pour it over the tea bag.
5. Wait until its reasonably strong, and remove the tea bag.
6. Put some milk in the tea, and drink it.

Figure 5-8. General recipe for making tea (from Figure 2-3).

This distributed structure explains much about how we recall our knowledge of activity. An essentially context-free description of an activity such as the abstract level of Figure 5-6 is as foreign to us as the precisely detailed description of Figure 4-7. Consider again, however, Figure 2-3: the description one might give, were one to ask how to make tea. This Figure is duplicated here as Figure 5-8. Note that some actions in this description are abstract in nature (e.g. *get a tea bag*), while others contain much more context (e.g. *fill a kettle with water, put it on the stove* - not just *boil water*). When I use my general tea recipe, I automatically boil water in this manner; indeed, I just go and turn on the stove by habit, without even thinking about the intermediate intentions (i.e. boiling water). My habitual methodology of boiling water using the stove has created a strong normative constraint to simply begin

doing it that way. This constraint makes the alternative of beginning to boil water using the stove the obvious choice at that point in time.

If something goes wrong however, I have the knowledge behind this preference immediately available, and I can easily access the knowledge supplied by intermediate intentions. If knowledge beyond my preference is available (Say I have been informed the stove does not work, or that boiling water is already available) other constraints make these alternatives more attractive and the original preference is ignored. This background knowledge is associated with my intention to make tea and is simply not accessed much of the time, because the routine aspects of the intention usually suffice. The constraints specified by the routine core of an intention form the bulk of activity: they are ignored only when they are obviously inappropriate (they may have conditions that stop the normative constraint from being active when it should be ignored), when other constraints are stronger (some background constraint may override the default), or when the agent is for some reason consciously dissecting its knowledge of the activity.[68]

There will, of course, be cases where reliance on compiled knowledge will get the agent in trouble: this is not infrequent during the course of everyday activities. In my own experiences of making tea for example, I have on occasion turned knobs the correct way (for my own stove), only to find later that the element controls were reversed on the stove I was using. Similarly, if the element my kettle is usually on is burnt out, I find myself continuing to try to turn that element on, only later remembering the element is broken. Such minor failures are a natural and expected part of the landscape of everyday activities, as are the mechanisms involved in extracting oneself from such failures. The forgiving nature of everyday activities discussed extensively in Chapter 2 supports this behaviour just as the broad and time dependent nature of these activities requires it.

The use of normative and general preference constraints also explains the nature of the descriptions that result when we are asked to describe how we go about an everyday activity. When I describe how to make tea, those parts of the overall intention (Figure 5-6) that have strong normative constraints associated with them will default to further intentions immediately, and

---

68 The former two cases are common in everyday activity; the latter would occur only in detailed planning (beyond the reach of the range of improvisation shown in Figure 4-1), as well as in learning. Both of these aspects are beyond the scope of this thesis.

instead of listing *boil water* as a particular step, I automatically move to my intention to boil water and look at its basic activities. Other steps may not have strong preference constraints (e.g. *get tea*) and so in a description they are listed in their general form. Despite the fact that in moving to further intentions, I consciously overlook one or many intentional levels, the point is that those levels *are* maintained in my knowledge representation: I could not otherwise adequately function when the routine did not fit the intended environment perfectly, or encountered unexpected errors or conditions. I could *without any effort* generate the complete internal tea-recipe of Figure 5-8 using the intention shown here, simply by following the normative and other preference constraints that usually apply[69]. This is exactly what happens when we consciously consider how we go about some everyday activity (i.e. attempt to describe in a linear fashion all of the actions we employ). These structures fit the phenomena displayed by everyday activities (Chapter 2) and the characteristics of improvisation (Chapter 4) precisely.

Thus, an intention structure serves as a link between the background concepts it entails and further intention structures that support the overall activity. Each alternative for action recommended by an intention may itself cause many further intentions to be recursively adopted, and thus each abstract step may translate into many individual actions (the grey blocks at the bottom of Figure 5-6). It is important to note that the actions resulting from an intention during its application are not cotemporal: they are the result of the gradual expansion of the intention, by committing to sub-intentions and carrying out actions over a period of time. The last concrete actions in a routine do not come into being until long after the first have been completed. They are only shown together in the Figure for the purposes of illustration[70]. *Because these actions are not cotemporal, the agent can, by making use of situation-specific normative constraints to hierarchical sub-intentions, make use of an intention in the same fashion as a universal plan. Normative constraints substitute previously acquired knowledge for deliberation, and as each new intention is adopted, and individual actions carried out, the environment will be altered and new constraints put in place. This in turn*

---

69 As well as the other constraints that make up the intention, such as requirements and expectations.

70 Figure 5-6 should be interpreted in the same fashion as one would interpret a trace of a best-first search, where all nodes are shown to illustrate the algorithm, even though later nodes will not be generated until earlier nodes have been examined.

will affect how future parts of the intention will be applied, duplicating the purely reactive reasoning of a universal planner. However, this structure is much less constrained than a universal plan: because background knowledge is represented and can be used as an alternative to normative constraints, the structure is just as easily applied to situations that are less than completely routine, something far beyond the scope of universal plans. This structure thus supports both deliberative, resource-bounded reasoning and complete reaction, allowing its application to a wide range of activities.

The use of intentions as a mechanism for knowledge representation may on the surface seem similar to previous techniques, most notably hierarchical planning [Sacerdoti, 1974], and the plan-subplan-task distributed hierarchical planning methodology of [Evans et al., 1992]. However, this is if fact very different from both of these. A hierarchical planner could produce the sequence of actions at the bottom of Figure 5-6. However, as mentioned above, the actions in this series *do not exist cotemporally*. The particular chain of circumstances (including the external environment, past actions, and other present and future activities the agent is intending) dictates the particular actions that will result from an intention, and this will differ in every instance. A normative constraint may be followed one time, and more deliberative reasoning applied another, leading to a different expansion of some sub-intention. *We are in effect planning as we go along: we make use of our previous experiences (stored as highly interconnected intentions) in order to make moment-by-moment decisions that are both locally and globally coherent.*

This structuring and its employment is also different from hierarchical planning in other ways. Intentions are more than simply abstractions of actions: they are interconnected experiences that have been performed in the past, and are put together over the course of many episodes of behaviour[71]. They connect all the knowledge required to perform an activity together into one package. Because of the large number of interactions (all represented by constraints), the whole is much greater than the sum of its parts. The use of constraints also differentiates these structures from other network

---

71 In Section 3.4, I argued that semantic as opposed to episodic memory was absolutely essential to everyday activities. Intentions, being built up gradually over many episodes of behaviour, are an example of semantic memory: no individual episode of the activity is ever recalled; rather, each is assumed to leave its mark on the structure through some learning mechanism. Such a learning mechanism is far beyond the scope of this thesis.

implementations of plans (e.g. Sacerdoti's [1975] *Nets Of Action Hierarchies*) in that, as mentioned above, the constraints do not act as strict ordering mechanisms between actions: some constraints may be strict, while others may indicate only very loose preferences.

Intentions are far more flexible than any structure used in planning to date, including both classical and universal plans. This flexibility is due largely to the distinction made between compiled (plan) knowledge and the background knowledge behind it. This flexibility is demonstrated in a number of ways. First, intentions allow compiled knowledge to be used as extensively as possible given resource bounds and the familiarity of the external situation to the agent, and background knowledge to be used in situations where more extensive deliberation is needed. They are also flexible in that the constraints set in (temporally) earlier parts of an intention directly affect the application of later parts, due also in part to their distributed nature. Finally, intentions are flexible because they are strongly influenced by the constraints surrounding them, allowing many external aspects to be considered that are not incorporated directly into the intention itself. As mentioned earlier, background knowledge of cooking activities may recommend checking ingredients for making tea before starting the activity in an unfamiliar locale, for example. Likewise, the same basic intention is used for grocery shopping under normal conditions and when in a hurry. The manner in which external constraints can change the nature of intentions differentiates these structures from others used in reactive planning systems. Firby's [1987] RAP system, for example, supports plans whose execution methods are decided upon at run-time, based on perceptual information. The interaction of an intention and the environment through constraints is much more sophisticated than this, both in the hierarchical nature of intentions and in greater complexities that can be represented using constraints.

5.2.1.1. Applying Constraints in Intentions

The previous Section has shown that the power of intentions lies in their constraint-directed and distributed nature. When actually making use of intentions, constraints are applied in one of two ways. First, many constraints in intentions are applied *hierarchically.* Recall that intentions subsume one another, just as everyday activities do: boiling water is an everyday activity (and an intention), but is also part and parcel of making tea (another intention). Because of this subsumptive nature, knowledge about the subsuming intention (much of it in the form of constraints) must somehow affect knowledge of the subsumed intention.

An example of the hierarchical application of constraints is shown in Figure 5-9. Here, the agent has adopted an intention to make tea (the intention illustrated in Figure 5-6). An external constraint instructs the agent that it is to hurry in performing this activity. The intention to make tea will consist of many recommendations for actions, and many constraints (e.g. that the work area should be kept clean). As part of making tea, the agent adopts an intention of obtaining boiling water. When it does so, the constraints in the *boil-water* intention are interpreted within the overall context of making tea. The *hurry* constraint is active during boiling water, as are all the constraints particular to making tea. Thus, if the agent has not boiled water often before, cleaning up spilled water may not be a part of the routine. Cleaning up spills may, however, be a part of the more global tea intention or a part of the agent's background knowledge, and this constraint would act hierarchically to influence the agent during it's participation in boiling water within the context of making tea. Information other than constraints is also preserved in this manner. For example, the *boil-water* intention has a obviously high priority within the context of making tea; however, this high priority is modified by the low priority of the tea-making activity as a whole.

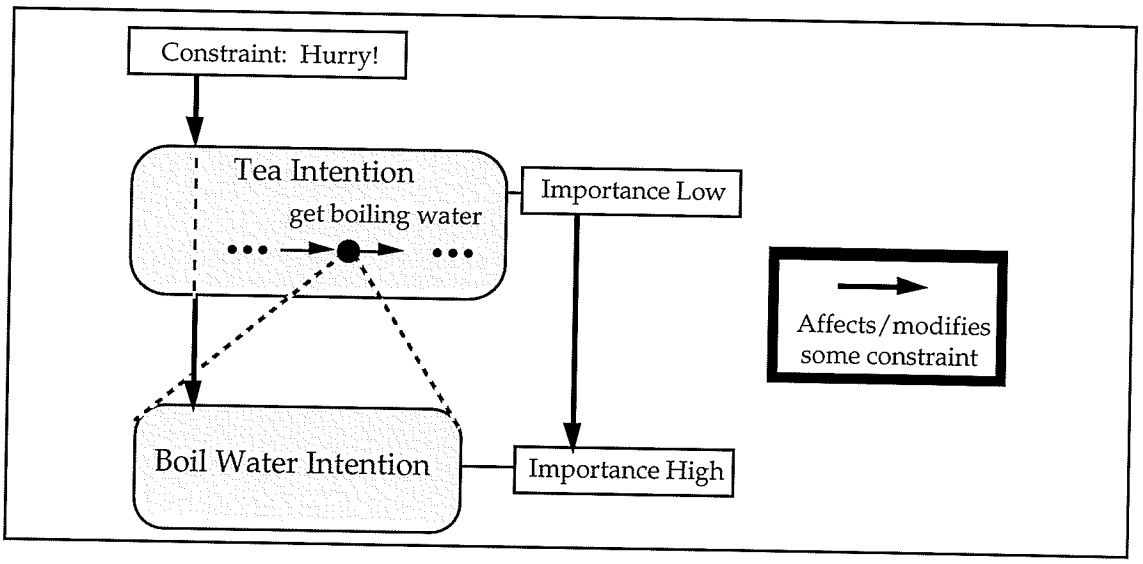

Figure 5-9. Hierarchical application of constraints in an intention.

In contrast to this use of constraints, constraints within an intention may also act *laterally*, and through connected background knowledge, represent many of the interactions that occur between everyday activities. An example of this use of constraints is shown in Figure 5-10. This Figure shows two different intentions that have been adopted by the agent. One is the same tea-making

intention used in the previous example, and is intended for the present time, while the other intention is the future intention to bake a cake. Now, these intentions will have many concepts in common (the kitchen setting, for one); however, for the ease of illustration, the background concepts belonging to each intention are shown as separate from the intentions themselves.

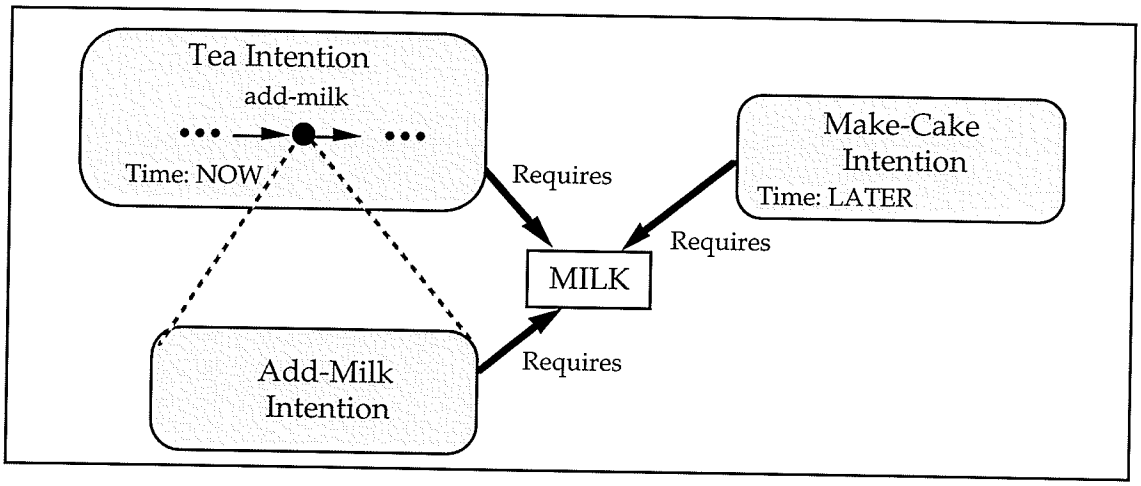

Figure 5-10. Lateral application of constraints in intentions.

In the tea-making intention, one of the necessary steps is adding milk to the tea (this is necessary when putting myself in the role of the tea-making agent, but may not be necessary for everyone). The agent has adopted a subsumed intention to do this, and one of the background requirements of this intention is having milk. Now, this is clearly also a requirement of the future intention of making a cake. The two requirement constraints, from two separate intentions, illustrate an interaction and are used to guide the agent rationally through these interactions. If there is only enough milk left for the tea intention, the direct connection to the cake-making intention will make the agent aware of this fact. Deliberation can then pursue, in order to determine which is the priority. Either one of the intentions could be abandoned in favour of the other, another intention could be adopted to obtain more milk, or the milk could simply be used for the immediate intention, postponing the problem of a lack of milk to a later time. Again, these decisions may involve extensive deliberation, or if the agent is extremely experienced with this particular interaction, a constraint may exist that prefers one over the other in a universal-plan-like manner. The choice of which method to use will depend on the amount of knowledge available and the context (active constraints) in which each intention exists.

5.2.1.2. Compiled and Background Knowledge in Intentions

Throughout this Section, I have emphasised the combination of compiled and background knowledge that make up intentions. However, thus far, virtually all discussion has been toward demonstrating the utility of preference constraints to resource-bounded reasoning, and demonstrating the ability of intentions to subsume universal plans. It remains to be seen how background knowledge comes into play, and how it is integrated with more highly-compiled knowledge such as the aforementioned preference constraints. Before moving on to the other types of knowledge that are required for improvisation, it is worthwhile exploring a specific example that emphasizes this combination.

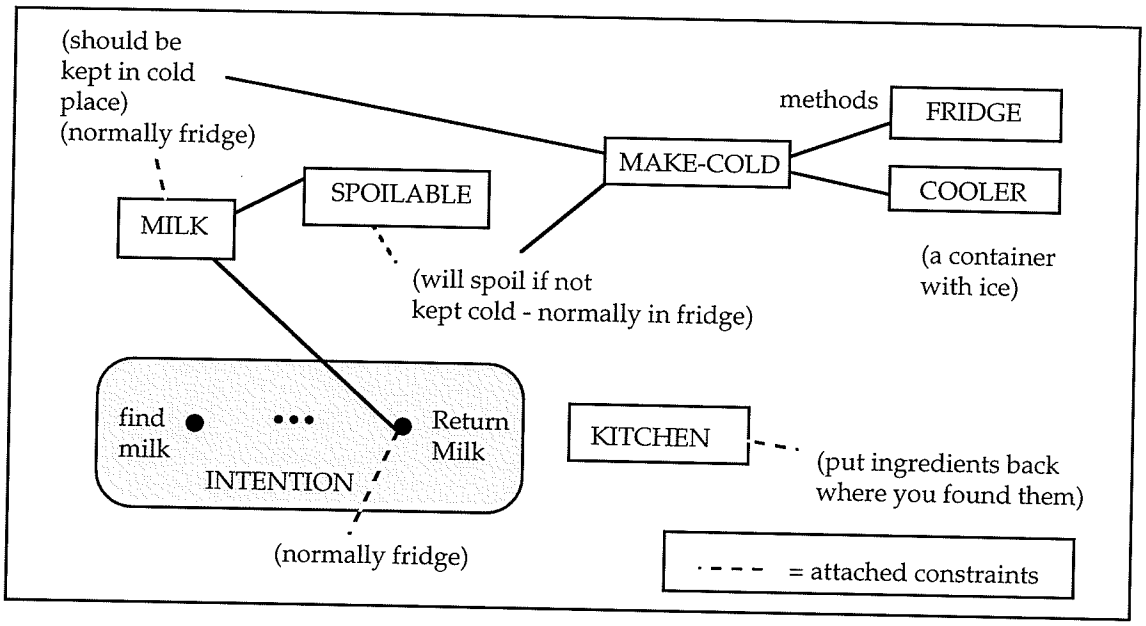

Figure 5-11. Making use of compiled and loosely associated knowledge.

This example is once again taken from the tea-making intention of Figure 5-6. Figure 5-11 depicts some of the knowledge involved in a portion of that activity, that of adding milk to the brewed tea[72]. Now, part of the routine for adding milk to tea is putting the milk away afterward. Returning the milk to the refrigerator is one action recommended by this intention (an obvious

---

72 This activity is not actually shown in Figure 5-6, for reasons of space. It immediately follows the *brew tea* activity.

normative constraint). However, there are cases where this preference is irrational: when the refrigerator isn't working, for example. The constraint to return milk to the refrigerator must contain a condition that the refrigerator be working. Realistically, this condition would rarely if ever be a checked: a human agent in this case would make the assumption that the refrigerator was working. The condition is present, however, so that if the agent did know that the refrigerator was broken, the constraint would be ignored. If the refrigerator is not working, the agent must move beyond the routine, and begin to examine the knowledge behind it.

Figure 5-11 illustrates the various pieces of knowledge (both closely and loosely connected to the intention) available to the agent in such a situation. The agent's *milk* concept is connected to the intention, and has a constraint that milk be kept in a cold place (this constraint advocates an action to put the milk in a cold spot, with a default of the refrigerator). The agent can also recall that milk is *perishable*, and that such objects must be kept cold. In both these cases, a refrigerator is the default (another preference constraint). The concept of *making something cold* is connected to both the *milk* concept and the *perishable* concept. This concept provides alternatives for the default which can be used to generate additional alternatives for activity[73].

The important thing to emphasize in this example is the dependence on memory: reasoning takes place through memory links that allow new concepts to be recalled. An agent can gather constraints by sifting through its semantic memory, and the more obscure the connection, the further it must go to find it. The difference between an obvious constraint and one that takes time to recall is the distance from the routine aspects of an activity in the agent's memory. This is also evident from previous examples. In the situation depicted in Figure 5-9, for example, during the course of boiling water the constraints attached to that particular routine are more obvious to the agent than the more general constraints associated with making tea or the background knowledge beyond that. Again, this is intuitive in the nature of everyday activities: the more obvious some constraint, action, or other piece

---

73 For the purposes of illustration, there is also background knowledge in Figure 5-11 that could guide the agent the wrong way. A general kitchen concept (whose nature will be described shortly) provides a constraint to place ingredients back in their original locations when cooking, giving rise to an alternative to put the milk back in the (broken) refrigerator. Somehow the deliberative component of an improvising agent must disallow consideration of such information.

of knowledge is, the more highly compiled and easily accessible it is. Compiled constraints (e.g. a normative constraint to simply put the milk in the fridge automatically) are easily accessed through their direct inclusion in the core of the intention, while the knowledge involving specific aspects of the activity is more distributed and less easily obtained. In improvisation in everyday activities, this external knowledge is just as crucial to coherent behaviour as the compiled knowledge that lies at the core of an intention. This background knowledge takes several forms. In the above example, we have already seen one: the idea of a conceptual setting that contributes constraints encompassing the agent's knowledge of the environment around itself. This and other types of background knowledge are described in the following Sections.

5.2.2. Settings

Whenever an agent is performing an everyday activity, it does so within the bounds of a given *setting*. In the tea-making activity described in Section 5.2.1.2, the activity was performed in a kitchen, and the fact that the agent was in a kitchen contributed a constraint influencing the agent to put any ingredients it used back where it found them. In the course of everyday activities, the setting in which they an intention is performed can contribute a great many constraints influencing how the intention is to be interpreted, in much the same way as a stage setting influences an actor. Being in a kitchen automatically predisposes one against many activities: anything that would be unhygenic, for example. It also influences one to be concerned with certain aspects that would be of less concern with in other settings: hygiene, as mentioned above; orderliness; even how careful we are of our actions. This is even more obvious in other physical settings. Consider the way one would walk in a china shop as opposed to on the street, or the way one might eat in an expensive restaurant as opposed to in one's own dining room.

In addition to such physical settings, an improvising agent is also strongly influenced by abstract settings that are usually independent of physical locale. The fact that one is in a public place contributes many constraints on behaviour that are not applicable (or are much less significant) in private. Even when in the presence of others, who those others are, the roles they occupy in relation to the agent itself, can further influence behaviour. Such social roles provide extremely strong constraints on human participation in

everyday activities. Even a simple act such as eating a meal can have a labyrinth of constraints that must be followed for social acceptance in many cultures [Visser, 1991][74].

In order to include these kind of influences, settings are an important part of knowledge representation within this architecture. A *setting* is a package of constraints, grouped with a condition describing obvious attributes that indicate the presence of a certain setting. A kitchen setting can be recognized by any of a number of physical objects that are usually found in a kitchen, such as stoves, counters, cupboards, etc. A more abstract setting will also generally have obvious conditions: the presence of other agents, for example, or recognition of their specific roles. Once activated, a setting contributes many individual constraints that serve the same purpose as any others in activity: they may guide, inhibit, or influence control. As a collection, however, each setting also functions as one large constraint, in that the conditions for the activation and lifetime of all constraints within the setting will be identical because of their origin.

5.2.3. Concepts

As discussed in Section 5.2.1, while intentions are the most significant form of knowledge representation in improvisation, the boundaries between intentions are difficult to distinguish due to the number of concepts two intentions will have in common. Because of this, the concepts that represent the background knowledge in intentions will represent the bulk of an improvising agent's knowledge. The interconnected and hierarchical nature of the conceptual knowledge required by the improvisation process is naturally suited to an object-oriented knowledge representation mechanism, such as Minsky's [1981] frames.

An example of object concepts organized in this manner is shown on the left side of Figure 5-12. Higher-level concepts both generalize knowledge contained in lower level concepts and through inheritance, provide a means of sharing information among lower-level concepts. Thus the *food* concept

---

74 Few examples of constraints are more obvious in everyday activities than those of social etiquette. The extremely complex rituals that culture dictates for even the simplest of everyday activities are often heavily underestimated. Visser [1991] illustrates many examples of constraints that must be followed when performing the simple act of eating dinner, and moreover, how strongly these constraints vary from culture to culture.

shown in Figure 5-12 describes general information about food, common to all examples underneath. Also as shown in the Figure, any number of intermediate level concepts are also possible: *food* has a subcategory representing those types of foods that are ingredients in food items, but in themselves are not edible. Lower-level concepts can also override general information, allowing high-level concepts to provide defaults to lower-level concepts[75].

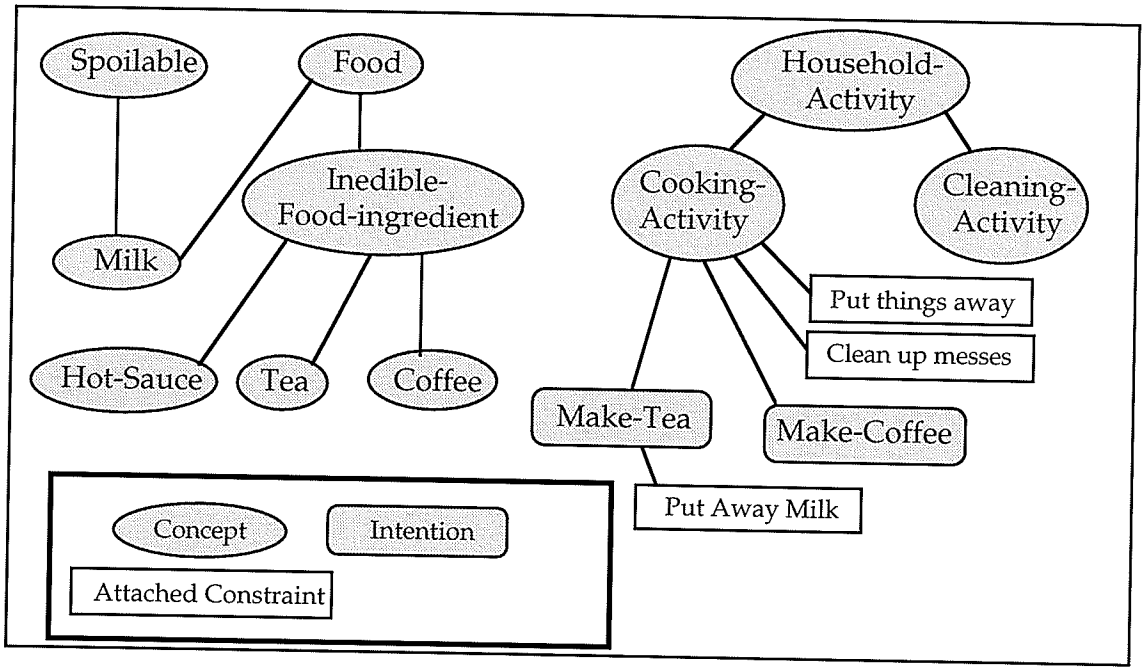

Figure 5-12. Multi-Hierarchical organization of concepts (Left:object example; Right: intention/activity example)

Because of the high degree of interconnection between the knowledge used in improvisation, representing the set of concepts involved in an improvised domain requires multiple inheritance. That is, each concept must be connected to more than one higher-level concept. Thus *milk* is described as a *food*, but is also *spoilable*, allowing information (including constraints) to be inherited from each. This could also be represented as a separate subcategory of food (just as *inedible-food-ingredient*). However, having a separate higher

---

75 These are all standard features of object-oriented representations, and all are described in [Minsky, 1981].

level concept (as shown in the Figure) allows the knowledge associated with something being spoilable to be shared with lower-level concepts that are not foods: that is, it eliminates redundancy. For the purposes of this architecture, the choice of how to represent such concepts is irrelevant. Constraints are not shown in this particular example, but high-level concepts can also contribute constraints toward the use of lower-level concepts in improvisation. A general utensil concept, for example, may abstract knowledge about spoons, cups, saucers, etc. This concept could contribute constraints such as requiring that all the utensils of one type match when used in a social setting, or that they be clean before using.

Objects are not the only types of concepts that are organized in this manner. Activity knowledge forming the background for intentions also has such a hierarchical structure. An example is shown on the right side of Figure 5-12. Here, the tea intention that has been used as an example throughout this Chapter is organized along with a make-coffee intention (for the purposes of example) under the umbrella concept of *cooking-activities*. The *cooking-activity* concept provides additional information about the activities it encompasses that may or may not be contained directly in the intentions below it. For example, the *cooking-activity* concept contributes a constraint that any messes should be cleaned up, and that ingredients used in the cooking process should be put away. As described in Section 5.2.1.2, the *make-tea* intention itself also contains a recommendation (constraint) that the container of milk used in the tea-making activity be replaced in the refrigerator. In this case, this can be considered a specialization of the more general constraint to a particular situation. This is also the case in general: higher-level concepts can make recommendations that suit a wide range of activities (e.g. clean up messes); by the same virtue, however, these recommendations will be much less situation-specific than any concepts below that level.

For activities in particular, it is intuitively important to improvisation to have this hierarchical organization as distributed as possible. When I make tea, for example, knowledge of activity outside of my tea routine does not come from one single activity concept underlying it. If it did, I would quickly be overwhelmed with a great deal of redundant activity knowledge the moment I consulted it. Rather, it is more appropriate to envision such knowledge distributed over a much more extensive hierarchy: knowledge of *hot-beverage-making* tasks, for example, separated from other types of beverage-making knowledge, etc. This makes for more redundancy, but

provides a more easily contained and easily accessible organization for this knowledge.

Inheritance through these hierarchical concepts may supply much more than constraints. In the case of intentions especially, high-level concepts may contain information (linked directly to specific intentions) that may be of use during improvisation. Consider looking for tea, for example (a sub-intention that would be adopted during the course of making tea). This could be by default an instance of obtaining an item from a jar (assuming we have a canister of tea available): an intention that would consist of finding the container, opening it, extracting the item, and closing it. Many variations of this same activity (e.g. getting something from drawers, jars, closets, or cupboards) could be grouped under one high-level concept (*get-item-from-container*) that could contain not only the specific constraints common to all these more specific activities, but an intention from which most of the knowledge needed to complete the activity could be obtained. Lower levels could then make the intention more specific.

Even given the assumption of a fairly distributed hierarchy, there is still an extremely large amount of information that can be gained through inheritance from any one low-level concept. It would seem unwieldy to put such a knowledge representation structure to use in practice because of this. However, it must once again be realized that the same difficulty occurs in human participation in everyday activities. When I think of a concept such as tea, I can obtain an incredibly vast amount of previously stored information: everything from how tea is used to its geographical growing region and the history of its trade. These in turn will link to many related concepts spanning the breadth of my knowledge. Very little of this is directly applicable when I am looking for a tea bag. I stop myself from recalling all this information simply because I do not invest the time necessary in traversing my memory for all this knowledge: other things occupy my mind long before I would reach such knowledge, or inclination stops me where knowledge is no longer particularly relevant. So it goes with everyday activities. The more common the knowledge to a particular activity, the closer it is to the core of the routine for that activity. The most crucial knowledge is stored in the core of the routine itself (e.g. experiential preferences for one action over another), while more common knowledge is directly connected to the intention itself, and less common knowledge is further away in the hierarchy. By the very definition of an everyday activity, the more crucial or common the knowledge, the easier it will be to obtain.

This Section has provided only a general description of the knowledge structuring necessary for the vast interconnected background of concepts necessary in everyday activities. I have described only the general requirements for knowledge structuring dictated by the characteristics of everyday activities laid out in Chapter 2. Over and above these requirements, there are many knowledge representation difficulties that must be solved to deal effectively with knowledge of the volume and variety necessary to support everyday activities. The difficulties in representation of concepts such as time, motion, and causation are well-known, and many approaches to such common sense concepts have been put forward, none of which have been completely suitable (e.g. [Hobbs et al., 1985; Hobbs and Moore, 1985; Pylyshyn, 1987]). Even accepting the argument that many of these theories of "commonsense reasoning" are developed for much more detailed philosophical cases than everyday activities demand[76], I must still acknowledge that a great deal work remains before AI can adequately represent this volume and variety of knowledge. However, as emphasized in Chapter 1, this is one of the difficulties with working in the field of intelligent agency: one project cannot answer all questions. From the point of view of knowledge representation, this research emphasizes the structuring of intentions and gives some requirements on the representation of background concepts, while recognizing that much work remains in the field of knowledge representation to adequately support this.

5.2.4. Reactors

Despite the wide range of applicability of intentions, there is still one aspect of everyday activities that does not fit the intention model described in the previous Section. Consider once again the tea-making example. If I pick up a cup, and the cup is hot enough to burn my hand, I will drop the cup either immediately or very nearly so. I do not appear to consider the burning sensation and consciously adopt an intention to drop the cup. Rather, quite the opposite occurs: there is an immediate desire to drop the cup, and I have to consciously adopt an intention to resist the urge to do so long enough to

---

76 Many of these theories of commonsense reasoning are far more detailed than any agent operating in the everyday world would require. As an example, one Chapter of [Hobbs et al., 1985] describes a "naive" theory of reasoning about shape, one axiom of which (describing the legal ways in which one object could move while still touching another that remains in place) occupies approximately one-third of a page of formal logic. Clearly everyday activities are more naive than this.

move the cup to a supporting surface. Even this intention can only be adopted temporarily: eventually the burning stimulation will override any conscious effort on my part. This type of activity corresponds to the fuzzy boundary between plan-based actions and simple unconscious reactions, described in Section 4.2. Such reactions are not part of cognitive activity in that we plan ahead to perform them or even think about what an appropriate response would be. However, they are absolutely required to support intentions, simply because there is no boundary line we can draw between plan-based behaviour and reactions.

Behaviour of this type is supported in this model through structures known as *reactors*. A reactor is a piece of knowledge consisting of a stimulus and a response: the stimulus portion of a reactor acts as a trigger for the knowledge structure, while the response consists of recommendations for activity. Reactor structures are not intended to be extremely sophisticated (as universal planning architectures would have them be), or even to play an especially significant role in behaviour. They are intended to represent suggestions for activity below a conscious level. Reactive responses are intended to be general: in the above example, for example, the reactor might monitor heat, and respond by suggesting moving the portion of the body that the stimulus is applied to away from the stimulus. The responses of reactors are also *suggestions* for activity, just as those from intentions. A reactor such as the one described above may recommend putting down the cup with some degree of intensity. If this intensity is strong enough, it may immediately override all other possibilities, effectively short-circuiting deliberation and moving right to action. Touching a very hot cup is of this variety: the suggestion is strong enough that action occurs immediately, superseding any deliberation. If the suggestion is not this strong, it becomes another of possibly many other suggestions that deliberation must sift through to decide on an appropriate action. Deliberation will be explained fully in Section 5.3.4.

Reactive structures such as these have been used previously in AI planning systems, most notably in the Phoenix architecture [Cohen et al., 1989]. Phoenix is both a simulator[77] and a set of agents that can be used to fight simulated forest fires. During a simulated fire, a central agent directs several bulldozer agents to cut fireline around the fire. As mentioned in Section 3.3, each Phoenix agent consists of a cognitive (deliberative) component, and a

---

77 The simulation aspects of Phoenix are described in Chapter 6.

reactive component. The cognitive component expands and applies stored plans, while the reactive component consists of a set of condition-response pairs that represent the agent's complete ability to react during any activity. The reactive component is run once every time step within the simulation, and causes the condition components of all reactions to be checked. if any of the conditions of a reaction are found to be true, the action is followed. For example, bulldozer agents have internal reactions to cause them to turn if they are heading too close to the fire.

The reactors used in this architecture are much more sophisticated than those of Phoenix. When a reactor is activated, the potential action it recommends is not necessarily immediately followed: it is input to a deliberation process along with any other potential actions that constraints dictated by the environment or the agents activities put forward. In most cases, reactions will be overwhelming in intensity (such is their nature in everyday activity, as reflected in the burning cup example), but if they are not, some intention may be adopted to temporarily stay the reaction. For example, if the cup was not burning very badly, I might be able to move to a countertop where I could set it down before the urge to drop it became overwhelming.

Reactors can also contribute constraints in addition or as opposed to immediate actions, and may thus serve to change the agent's attitude toward an activity or some part of its environment, thereby affecting activity indirectly. Finally, reactions in Phoenix are linear: there is a single set, all of which may be available at any point. Here, reactors may be part of the basic structure of an agent, but may also be attached to intentions, concepts, or settings, thus supporting activity- or setting-dependent reactions. Reactions are also relied upon much less in this architecture: in Phoenix, reactions were the only means of timely activity (that is, the only means outside of expending great cognitive effort to expand and flesh out a plan). Here, the intentions on which improvisation relies can be used with extensive deliberation as well as with very little deliberation, thus relegating reactions to a secondary position.

5.3. Waffler: A Constraint-Directed Architecture for Improvisation

> When we mean to build,
> we first survey the plot,
> then draw the model;
>
> – Shakespeare, *Henry IV, Part 2*

The previous Sections of this Chapter have focused on refining the abstract conceptualization of improvisation described in Chapter 4 to describe the concrete requirements of knowledge representation for an improvising agent. We have thus moved from being able to describe what an improving agent does in a colloquial sense (applying routine and background knowledge as appropriate) to a knowledge-based conceptualization of the events that occur during improvisation (allowing constraints from intentions to influence the agent's choice of action; supplementing this knowledge with further constraints on activity contained in conceptual knowledge linked directly or indirectly to intentions and obtained over time).

The remainder of this Chapter describes an architecture for applying constraints (through use of intentions and the other knowledge structures described in Section 5.2) to guide improvised activity. This architecture has come to be known as *Waffler*, after a colloquialism for improvisation[78] and the historical use of the agent termination "-er" in AI planning systems and languages.[79]

Waffler agents accomplish everyday activities through the application of intention structures. They recall these intentions from memory, along with conceptual structures describing background knowledge associated with these intentions. The structures in the agent's memory provide highly interconnected constraints describing the restrictions imposed by the agent's desires for activity and the environment around itself, and the agent employs the recommendations of these constraints during deliberation to make appropriate moment-by-moment choices for action. Waffler agents operate over time: they gather information and make arbitrarily timed choices for action. At any point in time, they can choose to act on the basis of the constraints they have recalled and examined thus far, or they can extend

---

78 Thanks to Vincent Wright for introducing me to the term *waffling*.

79 e.g. *PLANNER, CONNIVER, HACKER*, etc. [Barr and Feigenbaum, 1981; Cohen and Feigenbaum, 1982].

deliberation, allowing more information to accumulate (and potentially, opportunities to act to pass by). Choosing to act includes not only commitment to physical actions, but also to cognitive actions, such as adopting or abandoning intentions.

The primary characteristic of this process is that of *recall*. Knowledge in everyday activities is by definition common and mundane. It will be heavily interconnected, and as a result dealing with everyday activities is a matter of recalling the most significant constraints on the agent's activities at the time. As emphasized in Chapter 4, this is not the same as universal planning. The key to improvisation is to apply the constraints dictated by the routine when able, and to use the more extensive, less directly applicable, and less easily accessible background knowledge both to guide the interpretation of the distributed routine (i.e. emphasising which parts of an intention are the most important) and to apply directly to find appropriate alternatives when the actions recommended by the routine are inappropriate. At any point in time there will be intentions that will recommend routine actions, constraints in those intentions that recommend alternatives, and a vast network of highly-interconnected conceptual knowledge, each of which may supply further constraints (that may in turn support or refute existing alternatives, or give rise to entirely new possibilities for action).

The key to improvising is in selective recall of this highly-interconnected constraint knowledge: allowing constraints to guide the agent's activity toward certain alternatives at the appropriate times. The previous Section emphasized that this constraint knowledge is highly distributed between the core of the agent's intentions, external concepts such as the agent's setting, and a vast network of background concepts. The agent can recall any (or all) of these, given enough time, but is restricted by the dynamic nature of the environment it inhabits. An agent may spend a negligible amount of time in deliberation and be able to recall only the immediately available constraints on activity recommended by it's immediate intention, or may spend further time and have the ability to access deeper memory structures. Due to the nature of everyday activities, the most commonly-experienced alternatives and the most commonly applied knowledge is more heavily compiled and exists close to the core of the intention (as described in Section 5.2), while less commonly applied knowledge is buried further down in the agent's network of concepts. It is precisely this reliance on the recall of heavily interconnected knowledge Minsky [1986] emphasizes:

> Thinking in the human sense is facilitated by the well-connected meaning structures (neurons and sets of neurons) that let you turn ideas around in your mind, to consider

alternatives, and envision things from many perspectives until you find one that works.[80]

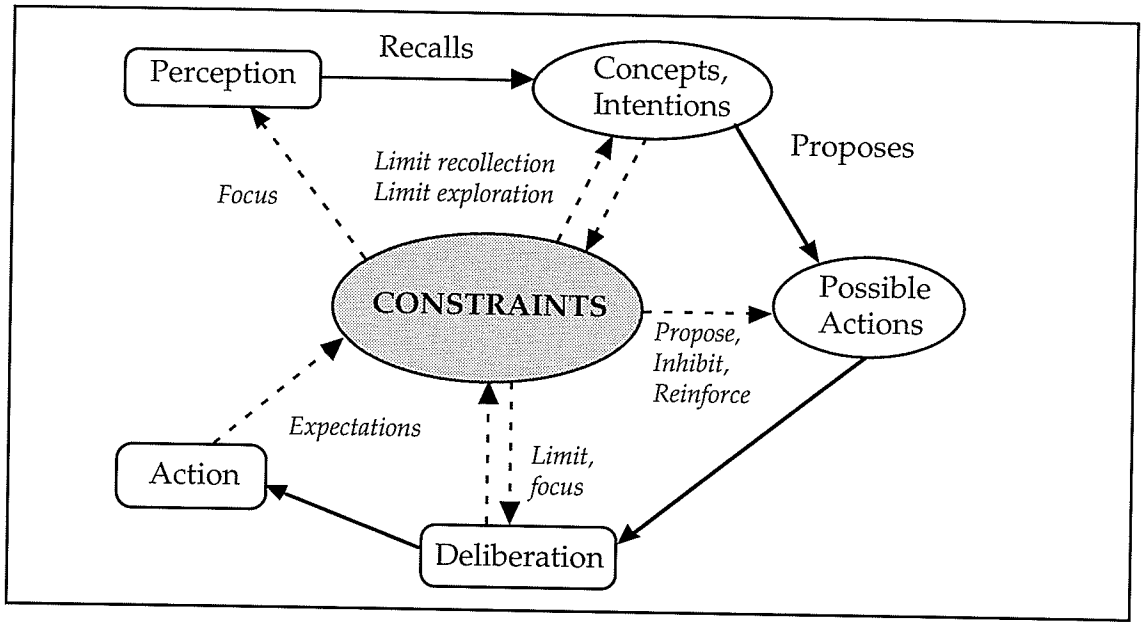

Figure 5-13. Processing involved in constraint-directed improvisation.

A high-level conceptualization of the processing involved in a Waffler agent is illustrated in Figure 5-13. The agent's perceptions (as well as its internal desires) trigger the recollection of concepts and intentions. These provide an interpretation of the environment around the agent, in light of the agent's pending activities, through a pool of constraints. Some of these constraints will directly recommend or discourage some particular action (including cognitive actions such as adopting or abandoning intentions). Other constraints may trigger further recall of knowledge structures, while still others will operate at a higher level. These higher-level constraints provide knowledge-based control over the entire process: they may limit the agent from recalling more concepts from memory than are absolutely needed, restrict deliberation in situations where prompt response is necessary, and may also mediate between lower-level constraints, among other functions. Each time the agent recalls some concept or intention from memory, more

---

80 This kind of conceptual recall does not have to be based on a neural implementation, although it certainly is in the case of humans, which is the perspective from which Minsky writes.

constraints will enter the mix. These new constraints may provide further restrictions, or may relax or override existing constraints. These constraints may be entirely contained in the recalled concept, or may be a result of interactions between concepts and intentions that have been recalled previously. Constraints will leave the pool when they are no longer applicable or when their lifetime has expired. At any point in time, however, the agent will have recalled a number of intentions and associated concepts, and will have the use of the constraints they entail as guiding knowledge.

The actions recommended by various constraints are deliberated upon, and as already indicated, the agent may choose to act whenever it so desires. The constraints active from intentions and the environment around the agent will dictate the length of its deliberations. When an action is chosen and carried out, it will have a set of expectations associated with it, which are expressed as constraints and serve as a focus for the agent's perceptual functions, directing the agent to focus its attention on areas or objects of interest with regard to its activities.

At first glance this may seem an unwieldy approach, given the amount of knowledge everyday activities have previously been shown to involve, and the time constraints normally found on these activities. If dealing with all this knowledge were indeed an impossibility, however, it would be equally impossible for human improvisation to take place. If improvisation truly involved an extensive constraint-directed search process (as problem-solving agents in AI would normally undertake), it would indeed be impossible to deal with this amount of information in any reasonable amount of time. This is the reason everyday activities have been an enigma to AI for so long a time. However, the processing described in this Section is plausible for the same reasons that human improvisation is plausible: *all of this knowledge contains a great deal of structure that can be exploited during the course of activity*. It is voluminous, but at any time the agent is dealing with only a very small portion of the knowledge it could potentially encounter, and processes it in a satisficing manner. *The intelligence in this approach thus revolves around focusing on relevant knowledge and avoiding having to deal with the vast amount of knowledge the agent possesses that is not immediately relevant to its current activities.* The processes described in this Section exploit the same regularities in the environment that humans do, and the processes themselves are based on characteristics of human behaviour in everyday activities described in Chapters 2 and 4. Specifically:

- Much of the time, the agent's routine will suffice directly for its participation in some everyday activity. This means that there will not often be large volumes of external concepts to deal with. *The number of constraints in this pool at any time will be relatively small, and the structure inherent in those constraints makes them easy to deal with.*

- The knowledge with which the agent works during the performance of everyday activities is highly interconnected - it is easy to recall not only any intention previously used, but any background concept. This interconnection makes it easy to grasp connections between a new concept and current intentions. For example, the sight of boiling water when the agent is making tea will instantly satisfy the agent's need for boiling water. The connection is automatic. For the most part, *there is no extensive reasoning required to connect background knowledge to current activities.*

- Using the knowledge structures described in Section 5.2, a Waffler agent can gradually move from routine to non-routine knowledge. *There is no need to immediately attempt to recall every piece of information known about some object or activity.* Agents recall only what they need to be able to trust their judgement on the propriety of an action. This again keeps the number of concepts the agent is dealing with at any one time small and the effort required to cope with all these constraints low.

- The most appropriate choice of action in everyday activities is for much of the time an obvious one. The action will usually be part of the routine, or will otherwise usually have overwhelming support from some crucial constraint representing the world around the agent or an interaction with its other activities. *There will generally not be many difficult choices during deliberation, by the very definition of everyday activities.*

- Most of the interactions between everyday activities is across time. Everyday activities do interact with one another (e.g. the phone ringing while making tea, as described in Chapter 2), interactions such as these usually involve only a few simple constraints. The bulk of the interactions are interferences between one activity and another in the future (e.g. a shared resource), which can also usually be easily resolved based on the importance of each activity. *Interactions are usually not complex in terms of resolving them, again by the very common and mundane nature of everyday activities.*

- As discussed in Chapter 2, everyday activities have a forgiving nature; we make mistakes during the course of everyday activities all the time, and generally recover without serious mishap. *Serious errors are rarely a problem, once again by the definition of everyday activities.*

To summarize, the operations of a Waffler agent as shown in Figure 5-13 would be unsuitable for dealing with huge numbers of constraints in real time with no structure to them whatsoever. The extensive search required in such a domain causes the same difficulty to a Waffler agent that is faced by a classical planner[81]. As this dissertation has emphasised from the start, however, improvisation has very little to do with such environments. The vast majority of human activity does not display this unstructured nature. By the very nature of everyday activities, an agent will *always* possess this kind of structured knowledge, making an approach such as that detailed in this Section both feasible and advantageous.

5.3.1. The Waffler Architecture

Having described the basic processing involved in Waffler agents, I now turn to a closer examination of the components that make up the architecture itself. An overview of the architecture appears in Figure 5-14. This Figure is simplified for the purposes of emphasizing the relationships between components over the complexities of the components themselves. The architecture consists of four major components, implementing the specific processes required by improvisation (Section 4.5).

In this architecture, the semantic memory required for improvisation is divided into two parts. An agent's *long-term memory* contains the agent's complete store of knowledge and experience: all possible intentions, reactions, settings, concepts, associated constraints, and the numerous connections between all of these. This knowledge is structured using the mechanisms described in Section 5.2. As described above, when improvising an agent selectively recalls only a small portion of its overall knowledge. The area in which this selected knowledge is stored is termed the agent's *working memory*. This working memory holds the agent's current intentions, other

---

81 By the same token, however, a Waffler agent is no more unsuitable to such an environment than a classical planner. Given enough time to come up with a solution, the Waffler agent can simply keep searching its memory and eventually piece together a plan in much the same way as a classical planner would.

currently significant knowledge structures such as concept and setting information, and reactors. An important area within working memory is the *constraint pool*, which holds the constraints on activity contributed by these structures. During the course of activity, the only knowledge directly accessible to the agent (and therefore the only concepts, intentions, and settings that contribute constraints) is that contained in working memory. New concepts, intentions, and setting knowledge must be brought in from long-term memory as needed.

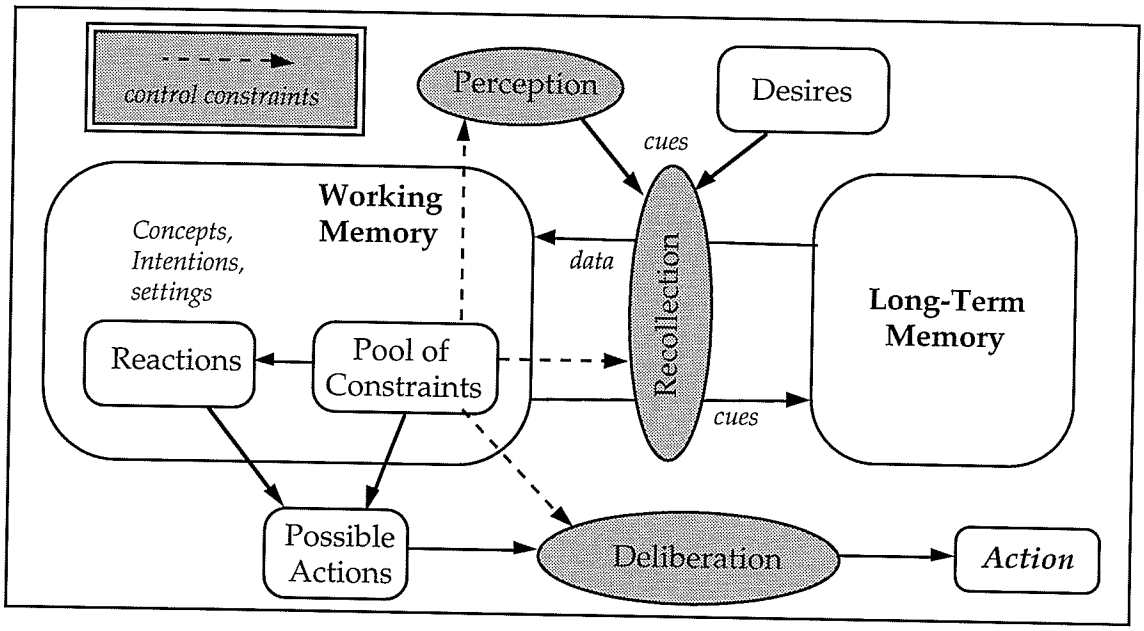

Figure 5-14. The Waffler architecture.

While working memory is not meant to be a model of human short-term memory, it will have implementationally-dependent physical limitations. Working memory represents the active workspace of the agent, where connections between pieces of knowledge in various activities can be realized quickly and the influence of constraints they contribute can be determined in a similar fashion. It represents a focus placed on all of the knowledge that could possibly be applied to the activity (long-term memory). As such, it will have physical limitations representing the amount of knowledge the agent can deal with at once. The agent can put as much knowledge as desired into working memory, but limitations will eventually require some knowledge to be lost (removed) as new knowledge is added. Because of resource bounds and the time-intensive nature of some decisions in everyday activities, it is often advantageous to keep the retrieval of information as limited as possible:

a great deal of new information could displace concepts in the agent's working memory that are relevant to guiding activity. In other cases, the reverse is true: we want to consider the ramifications of all new information on activity, regardless of the potential displacement of other concepts in working memory. The mechanisms to control the recollection and maintenance of information in working memory through the use of constraints are described in Section 5.3.3.

As described in Section 4.5, *perception* is also required by an improvising agent. The constraints currently active in working memory (i.e. the agent's expectation of the world around it) provide a focus for the agent's perceptual abilities, allowing them to filter out the knowledge most applicable to the agent's activities from the large amount of information potentially available. One of the major functions of perception is to stimulate the recall of knowledge from long-term memory. Perception, given knowledge of the activities in which the agent is participating and the activities in which it intends to participate in the future, generates cues for long-term memory which in turn cause the transfer of relevant structures to working memory. Perception can also trigger reactors in working memory, and allow them to make their contributions (recommendations for action or constraints). In addition, perception can also verify the satisfaction or violation of constraints, allowing the processing of intentions to proceed. For example, an agent may, after picking up a cup, expect the cup to be in its hand. This constraint forms part of the agent's perceptual focus, and the recognition of the cup in the agent's hand causes this constraint to be satisfied, and allows other activities to proceed.

An improvising agent also requires a *deliberative* component in order to decide between the alternatives put forward by constraints in the agent's knowledge base. The deliberation process illustrated in Figure 5-14 accepts the various alternatives proposed through intentions and through the other sources of knowledge described in Section 5.2, and commits to individual actions. This deliberation can take many forms. It may be as simple as considering a qualitative measurement of the utility of each of the potential actions available to it and selecting one, given some simple activity-specific conditions (*constraints*) as to whether waiting for more information would be likely to alter the agent's decision. Deliberation may also involve more complex measures of utility, taking various factors of each potential action into account. There are several factors common to deliberation in everyday activities however, and these will be discussed along with the nature of deliberation in everyday activities in general in Section 5.3.4.

In addition to these basic components, there is another component shown in Figure 5-14 whose influence on everyday activities has not yet been discussed: that of *desire*. Chapter 4 has extensively described both how an agent can be influenced to perform actions based on the constraints in its environment, and how the violation of one type of constraint (expectations) also directly motivates activity. However, I have avoided until now the issue of how the underlying, basic desires that motivate activity arise. The violation of constraints can explain why an agent may decide to pick up a cup it has dropped while moving it, and the action of constraints associated with previous experience can explain the decision to follow the various actions that contribute toward having tea. Neither, however, can explain the reason the agent *wants* the tea in the first place. Intuitively, the reasons behind such a desire could be a combination of many things: an interaction between thirst, cold, social custom, and caffeine addiction, for example. Although an agent needs reasons to act (that is, some means of generating high-level goals), I view the basic motivations behind desires to be both extremely complex and outside of the scope of this model of activity, and so I include no model of how the most basic desires arise in this architecture. Any top-level goal (that is, any desire for action that does not arise as a result of any previous action) will be assumed to be possessed by an agent without explanation. Some ideas on how a model of desire can be incorporated into this architecture will be described in Chapter 9.

Together, these components embody the behaviour shown in Figure 5-13: perception (focussed by existing constraints) arouses concepts in memory (as do the agent's current intentions), which contribute further constraints on activity (as well as the recollection of new intentions and concepts). These are interpreted by a deliberative component, which is also controlled by the constraints in existence at any point in time. Particular constraints may influence the agent to limit deliberation, for example. Such constraints may also have a more direct influence on one or more of the items the agent is deliberating upon. A constraint may indicate the presence of a particular situation that warrants waiting for more information before adopting some particular course of activity, or may restrict the amount of effort an agent is to take toward satisfying a particular desire. Both of these alter how alternatives are viewed during the process of deliberation. These control constraints also affect perception (in terms of focus and perceptual effort) and the process of recalling concepts from memory. These constraints and their effects on activity will be described in Section 5.3.4.

This architecture embodies the improvisational approach described in Chapter 4. A Waffler agent selects activities based on intellectual laziness, taking the path of least resistance by constraints. These constraints are attached to concepts, intentions, and settings in memory, which are made active when those concepts are in working memory. Constraints influence deliberation by their presence, as shown in Figure 5-1. The agent performs active search only when circumstances (represented by high-level constraints on activity) indicate that this is necessary. The use of the terms "intellectual laziness" and "path of least resistance" may serve to characterize this as a laissez-faire approach. In fact, it is quite the opposite. The constraints influencing activity serve to guide the agent toward decisions. The obvious action to take is by no means always the easiest or the simplest in terms of effort on the part of the agent. Many will be quite the opposite. *The principle of applying constraints in this manner, however means that the action selected will be the easiest in terms of following the agent's internal restrictions.* This is completely independent of the effort required by the actions themselves. For example, an agent may have a general constraint against stealing. Such an agent will find it far easier intellectually to go without what it needs or improvise some alternative than to violate this constraint, even though it may take more physical effort to do so.

Having described the general processes present in this architecture and the flow of information between them, we can now examine the control mechanisms necessary to integrate these processes. Given the description of Figure 5-14, there are several important aspects to control in this architecture: allowing constraints to make their contributions to activity (conceptually, the lowest level); managing the migration of knowledge in and out of working memory and ensuring that the most relevant constraints are available for the limited time in which an agent can deliberate; using deliberation to select from the actions the constraints in working memory recommend; and the process of allowing constraints in the agent's working memory to alter the methodologies employed during perception, memory recall, and deliberation, allowing the agent's control strategies to be tailored to fit the situation in which the agent finds itself. These are described in the following Sections.

5.3.2. Working Memory and Constraints

Section 5.2 has already described the knowledge structures that are contained in both working memory and long-term memory. What has yet to be explained is how information migrates from long-term memory to short-term memory and how the constraints common to all the knowledge

structures described in Section 5.2 operate in short-term memory to direct the agent toward courses of activity.

Constraints may perform numerous functions depending on their particular type (Section 5.1.1). However, the most obvious function (and that of most constraints an agent will possess) is to direct an agent toward or away from some alternative for activity. Each constraint may directly recommend a particular action or may directly refute a particular action. It may do so blindly (i.e. *do X*), or may contribute some level of intensity for the action for the purposes of deliberation. For the moment, I will call this level of intensity *utility*. For example, the agent may be making tea. The routine itself may directly contribute an alternative to adopt an intention to look for a kettle with which to boil water (defaulting to a particular intention, as described in Section 5.2.1) with a certain utility. If the phone should now ring, this will trigger a normative constraint (attached to the phone concept, which will be brought into working memory when the agent realizes the phone is ringing), which recommends an alternative of answering the telephone, also with a degree of utility. Other constraints in working memory may add to the utility of a particular action (enhance), or may subtract from it (inhibit). Recall from Section 5.1.1 that constraints themselves have a utility, a measure of their significance within the current situation. This is the same concept as the utility of an action, and the utility given or removed from a potential action is directly affected by the utility of the constraint that performs these modifications.

Reflecting the way that new objects we see and recall during the course of everyday activities are instantaneously integrated with the activities we have in mind, the influence of an active constraint in working memory is viewed to be incorporated instantly as soon as it becomes active. Constraints are viewed as parallel, active structures, with the ability to make their contributions as soon as they appear in working memory. Unless permanently resident in working memory, a constraint will be attached to one of the types of knowledge structures described in Section 5.2, and will become active immediately upon being moved into working memory. As described in Section 5.1.1, it is also possible for a constraint to have a physical set of activation conditions associated with it, over and above the condition of its attached memory structure being present in working memory. These conditions act as a trigger for the constraint, further indicating when the constraint should be able to contribute any recommendations it has.

The parallel nature of constraints means that a newly-recalled constraint can make its contributions immediately: any connections to concepts already in

working memory are immediately realized. It also means that connections between existing constraints and newly-recalled concepts are also realized immediately. This does *not* mean that an improvising agent has thousands of processors that are filled with constraints and allowed to run. This is not an active matching (search) process. Rather, it is a reflection on the structure of the agent's memory. Recall from the previous Section that the agent's memory (both long-term and working memory) is highly interconnected. A constraint enhancing or inhibiting a particular alternative should be viewed conceptually as *already connected* to that alternative. Thus the idea of constraints making their contributions known in parallel is a statement of a requirement of the memory of an improvising agent, rather than of some sophisticated processing mechanism as part of deliberation[82]. The contributions of particular constraints are instantly available due to the structure of working memory. Deliberation of alternatives involves reasoning about and comparing particular actions, and is another issue altogether.

As mentioned above, the idea of constraints operating in parallel via memory connections is inherent in how humans perform in everyday activities: the fact that this knowledge has been applied previously many times implies direct memory connection. However, we must recognize that there are limits to this. While I can immediately recognize the implications of a new concept on my current activities, my chances of forgetting to consider some important constraint depend directly on the number of activities in which I am currently participating (i.e. the number of concepts I have to concentrate on) and my focus of attention. If I am performing only one activity, the mental effort is not tremendous. However, this process has its limits. When confronted with fifteen activities, and integrating new pieces of information in an ongoing manner, I would be forced to write them down or risk missing some connections to activities that were not part of my current focus at any one moment in time. That is, constraints and their consequences can be activated in parallel, but this parallelism will always have a physical limit.

---

82 Such a memory is currently difficult to implement and manage computationally, although human memory is a representative example. The difference between improvising with a human-like memory and with a serial memory is simply time: it will take much more time to process the constraints in working memory in a serial manner. This is an implementational issue and will be described in Chapter 7.

This is the reason for the fundamental division in memory in this architecture into long-term and working memories, and for the limited size of working memory. The limitation chosen for working memory represents the limit of parallelism in that particular implementation of a Waffler agent[83]. Intentions occupy working memory, and some limit will eventually cause no new information to enter without forcing less important information to be removed. So when dealing with multiple intentions, there may be times when the agent must have only some of its intentions in working memory (that is, it must move less important or more temporally distant intentions back to long-term memory where constraints cannot act in this parallel fashion). This is a natural consequence of the volume of information and is common in everyday activities: we recognize interactions and consequences quickly, but cannot keep everything in our heads at once.

Though already discussed in Chapter 2, it should be restated at this point that having large numbers of intentions is not characteristic of everyday activities. For example, keeping a list of my business appointments and considering all the interactions between them in order to plan an optimal day would be something I would do using pencil and paper, and some detailed analysis. It is far beyond the realm of everyday activities. In everyday activities, interacting intentions are likely to be much simpler and smaller in number (e.g. reasoning about whether putting milk in tea now will spoil making something else requiring milk later). In the majority of cases, an agent's working memory will be large enough to handle the number of interacting intentions and their associated concepts common to the majority of everyday activities. This is certainly what occurs in human everyday activities. When the amount of incoming information overtakes the abilities of our limited mental focus we either suffer with the potential for errors, or we make use of memory aids (e.g. calculators, note pads, computers). That is, we deal with it as best we can within the framework of our everyday activities, or in a situation where this degree of satisficing is not tolerable, we move beyond what are normally considered everyday activities.

The assumption of such heavy connections made in the design of working memory is not at all out of touch with the current state of the art in AI. In fact, it is fairly conservative compared to many systems. For example, in

---

83 It is also possible to say that an agent's memory can reasonably handle matching N constraints in a given unit of time. Conceptually, it is simply easier to think of N as an absolute size of working memory.

Hammond's *Trucker* system [Hammond et al.,1990], agents receive requests to pick-up and deliver parcels at various point in a simulated neighbourhood. During the course of making deliveries, agents may pass by or near locations for pending requests. When this occurs, the agent can integrate the new request into its current agenda. After doing so, the system stores a reminder that it once had an order in the intermediate location, and if the original order is repeated, the agent will automatically recall that it once had a parcel in the intermediate location when it passes by, and checks to see if there is another. Such automatic recall is much more extensive and questionable than that proposed here: each individual episode of activity is recalled, even when it has only the most remote chance of being required[84].

The direct links (compiled knowledge) that allow a constraint to make its contributions immediately and in parallel can also be used in a larger sense. Consider the interaction of an external constraint such as *be careful* on an intention such as making tea. Now this constraint could be used at a low level, influencing the agent to choose one action over another or one intention over another. However, in the case of making tea I know beforehand how being careful would affect the activity: in addition to control aspects such as taking more time in making decisions, there are certain very specific aspects of the activity that are altered. I would check that I had all the ingredients I needed, for example; I would also be more careful in my coordination when pouring hot water, to ensure that I could react quickly to any spills. There are many other ways in which such a constraint would knowingly affect an activity such as making tea. If being careful were not a common aspect of making tea (that is, if the user had never been careful when performing activity), the best that could be expected is that the *be careful* constraint would contribute varying utilities to alternatives as they were considered. However, being careful is a common part of the activity, and so it should be expected that as a common aspect of the activity, the agent should be able to very quickly realize what overall effect being careful has on the activity. This is done in this architecture by allowing external constraints such as this one to have direct links to portions of an intention. Thus the mere presence of a constraint such as being careful can alter an intention, to bring out aspects that would normally not be considered and inhibit other aspects that might be normal in the ordinary course of the activity. There are many constraints in any activity that can be used in this manner: in fact, any

---

84 This system and its relationships to this research will be described further in Section 8.4.

constraint with which the agent has a great deal of experience. In tea-making and drinking, for example, *politeness* is another such constraint. The customs that being polite bring to activity are common and well-known, and can be easily realized with no inference or memory references because of this.

It should be emphasized again that the use of constraints in the manner described above represents compiled knowledge: the methods by which the constraints affect the intention are known and can be made without any conscious inference. *In no way is there any expectation that all constraints are processed in this manner.* An agent must have a great deal of experience with a condition such as being polite when making tea to have the constraint directly connected to those portions of the routine that it affects, in the manner described above. If the agent were less familiar, the constraint would not have such a direct connection and would instead influence the agent's behaviour through the methods described earlier: through adding or removing the utility of particular alternatives, and by allowing further alternatives to become available through more active search of the agent's memory structures.

5.3.3. Long-Term Memory Migration

The limited size of short-term memory makes it necessary to migrate items to and from long-term memory. However, the intentions that are the fundamental knowledge structure for improvisation have no concrete physical boundary: they consist of a routine and a huge collection of background information. Potentially, an intention could involve every single concept in the agent's long-term memory. Complete intentions (that is, the inclusion of all background information) cannot be moved into working memory at once. Similarly, a concept may contribute constraints, but so may its parents back through the conceptual hierarchy: too large a collection to hold in working memory at the same time. These are particular instances of a general problem: given one concept, how to know when to bring in concepts associated with it, and how to handle all this in an efficient fashion.

The answer to this problem lies again in the structuring inherent in the agent's knowledge of everyday activities. In any intention there are concepts that are vital to the intention and should be accessed often, and those that are used infrequently. In the infrequent cases, however, there are characteristics that identify the situations in which this knowledge is useful. These characteristics may sometimes be vague, but some guess will always exist.

These can be included with the portion of the intention in working memory, and can be used to decide when to bring in aspects of the intention that are more distant.

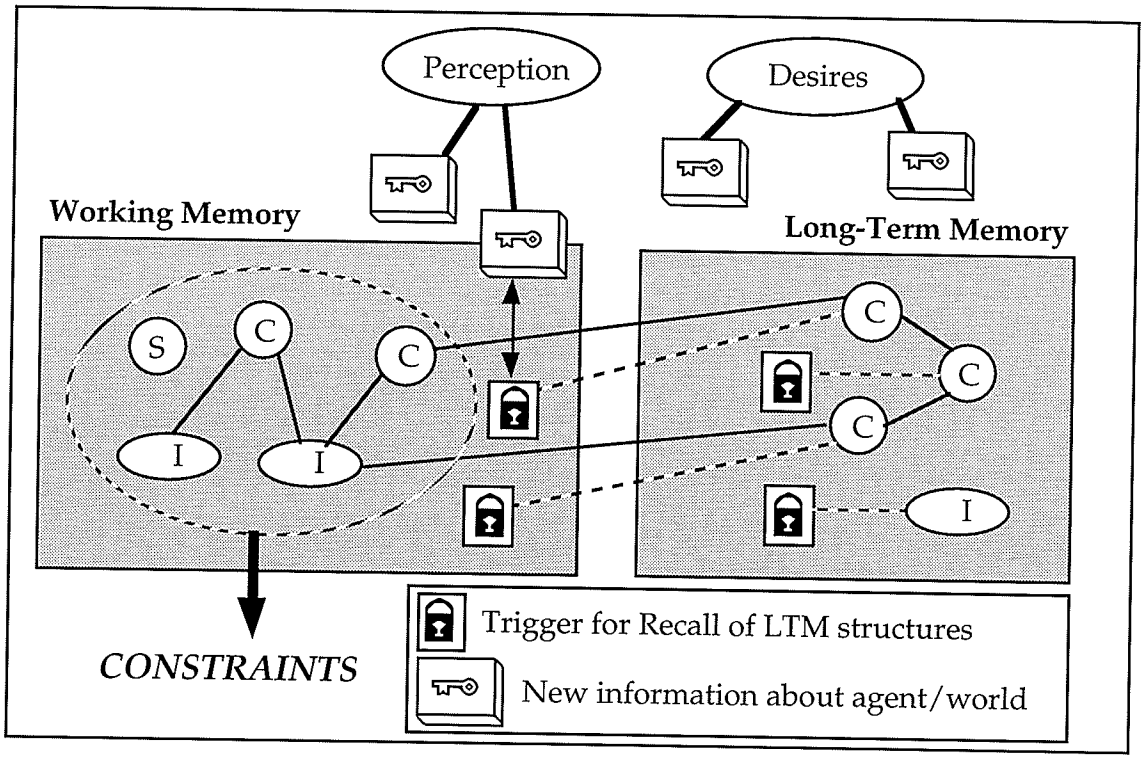

Figure 5-15. Details of memory recall.

A conceptualization of this process is shown in Figure 5-15. Here the agent has several intentions it has adopted, which are contributing constraints to its decision-making process. These intentions are not stored in working memory in their entirety: only the most crucial aspects are initially brought in: the routine itself and a few extremely relevant concepts. Those parts of an intention that are not present in working memory are represented by triggers: active structures that look for information indicating that the knowledge should be retrieved and stored in working memory. When new information is obtained (e.g. a new desire or a new piece of information obtained through perception), these are automatically recognized by the triggers present in working memory (i.e. all are processed in parallel) and the associated concept or intention is retrieved. This may in turn bring further triggers into working memory. This process allows information to be moved into working memory as needed. It also allows the smooth swapping of information when working memory space becomes limited: moving in one

concept may cause another to be forgotten, but its trigger remains and it can be recalled when needed in the future.

This conceptual method of selective information retrieval is useful any time an agent has to deal with a large volume of knowledge and the conditions under which various concepts may be useful can be defined. It is based on an approach originally used in the PIP expert system for diagnosis in internal medicine [Pauker et al., 1976]. In this system, a short-term memory (roughly equivalent to the working memory described in this Section) was used to hold hypotheses for the diagnosis of the illness of a presenting patient. Each concept in an associated long-term memory was given a set of triggers that would cause it to move into working-memory, along with any hypotheses that concept might in turn trigger. This would in turn cause more triggers (of concepts directly connected to the to those just moved into short-term memory) to be drawn into short-term memory, checking for signs that their associated concepts were useful. These new triggers would not necessarily represent less frequently used knowledge as is the case in everyday activities. Rather, in PIP this process is used as a method of making inference using the current hypotheses the system possesses. For example, a given set of symptoms might match the trigger for a particular medical condition, such as acute glomerulonephritis, which in turn would be connected to other states, diseases, or conditions related to it. Concepts with direct connections would not immediately be brought into working memory, but would allow their triggers to be moved in, giving the potential for further inference. This process is by no means novel; however the adaptation to everyday activities and the characteristics of this knowledge does have novel aspects.

Over and above what an agent automatically retrieves into working memory, it is also possible for the agent to actively search its long-term memory for some specific piece of information (a cognitive action). This gives the agent the ability to move beyond the concepts in its working memory and obtain less frequently used knowledge "buried" further down in long-term memory (as already described), with the added cost of a delay in retrieval. Once in short-term memory, constraints associated with the concepts contribute their knowledge in parallel as discussed above. This is not an active search process in the traditional sense, but rather a process of gradual recollection over time. This process is used often in making decisions for action: an agent may decide it does not have enough information to commit to one particular course of action, and may allow more time to access further concepts in long term memory that may contribute constraints to aid it in its deliberations. For example, a preference constraint may recommend alternative *X* over

alternative *Y*. If the agent wishes to wait, more concepts related to these decisions may be retrieved, giving further information (knowledge behind the constraint), which in turn may indicate stronger or weaker reasons for the alternatives in question.

The other side of migration between long-term and working memory is out-migration. As mentioned in the previous Section, working memory can become full, in which case some knowledge will have to be removed before any new concepts can be retrieved. When any knowledge structure is moved into working memory, it has a specific lifetime. This may in be a specific period of time, or it may simply indicate that the structure is to be resident as long as some other structure is needed. The lifetime of structures in working memory will be staggered, and as a result, working memory is cleared out on a consistent basis. More importantly, as intentions are completed, the knowledge structures representing them are removed, along with the external concepts connected to them. In spite of all this, there will still be occasions when working memory will overflow. In these cases, any memory management scheme may be used: old structures may be removed first; concepts removed first (leaving the cores of intentions), etc. Methods of applying appropriate memory management strategies will be dealt with in Section 5.4.

5.3.4. Deliberation

As has been emphasized throughout this Section, the majority of the actual work involved in improvisation is concerned with memory. Selective retrieval from memory allows the agent to recognize relevant constraints dictated by its environment, its pending activities, and its previous experience with activity. These constraints give rise to alternatives for action (both physical actions and cognitive actions such as adopting intentions), as shown in Figure 5-16, that must be deliberated upon. Recall however, that the choice of action in everyday activities is usually obvious, and the consequences of an incorrect choice are usually not dire. This is due (as described in Chapter 4) to the application of domain knowledge in the form of constraints. The really crucial part of this architecture, therefore, is in ensuring that constraints can contribute their restrictions and guide the agent toward relevant courses of activity and that constraints are made known and applied at the appropriate times. These are handled by the agent's working memory (Section 5.3.2 and 5.3.3) and the knowledge structures employed in improvisation (Section 5.1).

This does not mean that deliberation is insignificant: clearly significant and extensive deliberation will be rare given the nature of everyday activities.

However, deliberation between choices is virtually always necessary, and since it is viewed as operating in parallel with memory recall, the deliberation process has the added burden of deciding when there is enough information to commit to actions and when it should wait for for more support. However, the nature of the agent's deliberations is relatively straightforward, given the nature and consequences of everyday activities.

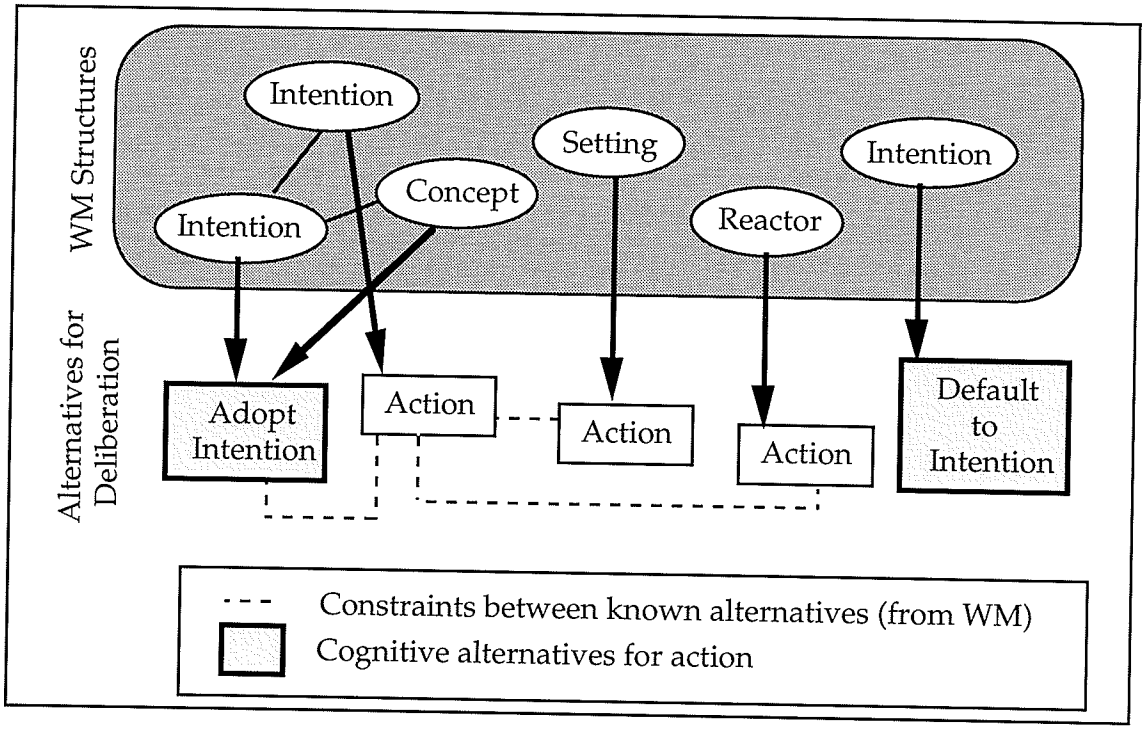

Figure 5-16. Generation of alternatives for deliberation

The most important concept in a Waffler agent's deliberations is *utility*. Utility is a basic measure of the "goodness" of an action, as defined by the constraints in the agent's working memory: obeying a constraint may add a particular utility to a potential action, for example. Such a measure could be qualitative, or a quantitative measure could be used. At it's simplest, deliberation involves the trivial process of selecting the potential action with the largest utility.

However, the fact that all the knowledge necessary to make an informed choice may not be available at once complicates this. At any given time, each of the alternatives available will have a given utility (e.g. those shown in Figure 5-16). However, if the agent waits to make its decision, more information will be processed by the agent's short-term memory, possibly

changing utility measures or introducing new alternatives. Waiting for more information may also invalidate other alternatives if there is a limited time-window for their availability. That is, deliberation is a *temporally dependent* process.

To deal with this, we must recognize the alternatives and their initial measures of utility for what they are: most of what will initially be available in terms of alternatives for action and their respective utilities is compiled knowledge: routine actions and preferences. These will, by the nature of everyday activities, suffice much of the time. Moreover, also by the nature of everyday activities, *the agent should have some knowledge of when this is not good enough.*

There are two measures through which this is defined in this architecture. Both, again, make use of the concept of a constraint. Primarily, the agent's deliberations operate with a *utility threshold,* a constraint representing the minimum level of utility required to be able to commit to an action. In situations where one or more actions with a high enough utility are available, the agent commits to the best. When no action is available, the agent waits until an appropriate choice arises. This utility threshold is dynamic: it can be raised or lowered by constraints in working memory. A constraint such as *be careful* , for example, can when activated not only contribute utility toward (or take utility away from) particular actions, but can also raise the agent's utility threshold, ensuring that the actions an agent commits to have more than an adequate amount of support. Similarly, if the agent has an existing utility threshold and no amount of waiting will give it something to do, it may lower that threshold and proceed as best it can.

Over and above this, there are also many times when the agent knows to be suspicious of particular alternatives for activity. The general cases can be handled by general constraints such as the *careful* example above: having an accident such as dropping a cup may cause such a constraint to become active, causing the agent to be generally more careful in its activities in the immediate future. The agent may also, however, have specific pieces of knowledge associated with concepts or constraints that allow it to be suspicious of particular alternatives. For example, if the setting is very different from the usual for a particular activity, this may indicate a need for more deliberation, and would subsequently raise the threshold required for a potential action to be acceptable.

Deliberation is inherently simple in everyday activities because most of the work involved in differentiating good alternatives for action from bad ones is

done by the constraints forming the agent's knowledge of activity and the world around it. When the guidelines provided by the agent's knowledge are weak, fewer or weaker constraints are provided, that do a less adequate job of making one alternative obvious, deliberation becomes more complex. That is, reliance on compiled knowledge is a substitution for deliberation (knowledge trading off with search) made possible by the nature of everyday activities. It is precisely when the situation in which the agent finds itself strays from the everyday that its guidelines become weak. The less experience the agent has with the particular variant on activity or setting, the weaker will its guidelines be, as the agent strays farther and farther from everyday activities (as per the spectrum of activity described in Sections 3.5 and 4.2).

In order to cope with less routine situations, we must recognize that there will be times when deliberation will get complex enough that a single utility measure will not suffice for adequate comparison. I wish to emphasize that I do not intend in this Chapter to bridge the gap between improvising and problem-solving (as discussed in Section 4.2). This dissertation addresses only everyday activities as I have defined them in Chapter 4. I claim only that which has already been stated: that as the agent moves away from everyday activities and toward problem-solving, the guidelines provided by its intentions (constraints) become weaker, and deliberation must compensate by becoming more extensive (in conjunction with a more extensive search of the agent's long-term memory for appropriate alternatives).

In order to allow more extensive deliberation, variations on utility must be provided. Utility represents, as mentioned earlier, the "goodness" of a particular action in a global sense, as defined by the constraints that support and refute the alternative. It is a currency-based measure [Minsky, 1986] that allows us, effectively, to compare apples and oranges. However, such measures allow wide comparison by hiding the very factors that make the items under comparison different from one another. Certainly, one alternative may have a greater utility than the other, but the agent is often more interested in the specific factors that make up this utility. In everyday activities, there are three factors that are especially significant:

- *Effort*. The estimated effort of some alternative is an important aspect of its overall utility. When comparing various methods of achieving the same desire, it is an important factor in utility when all other factors are equal. For example, say an agent runs of milk for its tea. Now, there are several obvious alternatives: not having milk, having some form of coffee whitener as a substitute, or obtaining milk. There are varying

amounts of effort associated with these (intuitively, they are listed in order of increasing effort). If the intention of having milk in its tea is not particularly important to the agent, however, the earlier alternatives become much more attractive than the latter.

- *Compatibility.* When several alternatives for an intention arise, not only may they differ in effort, they may differ in the degree to which they satisfy the agent's original intent. To use the above example, each of the alternatives satisfies the agent's desire to have milk in its tea to a different degree. Going and getting milk, while requiring more effort, would certainly satisfy the desire better than substituting coffee whitener, which in turn is better than nothing at all[85].

- *Timeliness.* The agent must also be concerned with the perceived timeliness of each alternative: how important it is to act quickly if it is to take advantage of the alternative.

Effort and compatibility are applied in a similar fashion as utility: thresholds (constraints) exist for compatibility and effort, and alternatives not meeting those thresholds cannot be selected as choices for activity. However, constraints on effort and compatibility are inherently different from utility. Because we reason about the effort and compatibility of alternatives for one specific course of action, the constraints the agent possesses on effort and compatibility are not global. Instead, restrictions for compatibility and effort are defined for a particular intention or desire. When an agent adopts an intention, the intention will have associated with it restrictions on the effort to which the agent should be willing to go to satisfy the intention, and the compatibility that is required. This effort and compatibility are modified hierarchically as further intentions are adopted. That is, as shown in Figure 5-9, each new intention contributing toward the same activity may modify these restrictions. As an example, consider the running-out-of-milk example mentioned above, and assume for the purposes of example that compatibility and effort are measured quantitatively on a scale of one to ten. Now, the intention of putting milk in tea will have constraints on effort and compatibility associated with it (and modified by higher-level intentions). Assuming the agent has knowledge that there is no milk in the house, an alternative would arise from this intention to adopt an intention to obtain

---

85 Although many coffee-drinkers would dispute this.

milk. The situation that results from adopting this intention is shown in Figure 5-17.

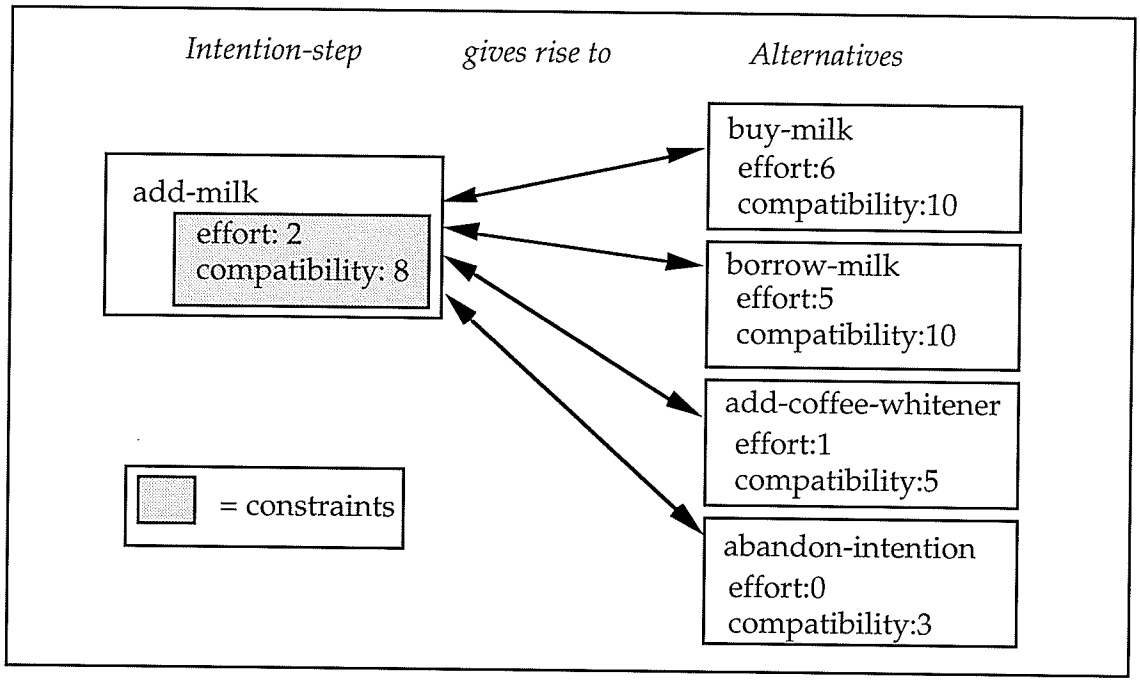

Figure 5-17. Constraints on effort and compatibility.

Through the processes described earlier in this Section, this intention would cause several alternatives to arise. Some of these would be common alternatives and would likely be stored in the intention itself (e.g. abandonment, adding coffee whitener) while others may be obtained through attached concepts (e.g. buying or borrowing milk may arise through information in the agent's *milk* concept). Each of these has an intuitive effort and compatibility, which the agent can use in conjunction with the constraints on these factors associated with the intention.

When deliberating in situations such as this, the agent is necessarily never working with complete information. Each possible alternative for action will have an intuitive estimate of utility and effort, for example, but these intuitive estimates will vary widely by context: going to the store for milk increases in effort greatly when it is late at night, for example, when few stores remain open and the agent may have to travel further. The only way

to ensure that information is as accurate as possible is to consult every concept in the agent's long-term memory to ensure that all possible affects are accounted for[86]. This would require a great deal of time in most cases, to say the least, and it would be an *extremely* rare situation in the course of an everyday activity for this to occur.

Now, in this case, there is no intention that matches the constraints on effort and compatibility. Each of these alternatives either takes too much effort or is not compatible enough with the original intention. The agent now has two alternatives available to deal with this situation. First, there may be constraints in addition to those shown in the Figure that allow the agent to choose one factor over another (e.g. not having milk is not as compatible, but takes less effort than the other alternatives; if effort is important at the time, this should take precedence). This will be elaborated upon in the next Section. This would allow a choice of action, but in most cases we cannot expect to be told that one factor is more important than the others.

In these cases, *time* is the factor the agent uses to make a decision. The more the agent delays its decision, the more accurate its estimates of effort and compatibility will be. If the agent were to wait awhile, more concepts will be retrieved to working memory and the measures of effort and utility associated with its current alternatives for activity may go up or down. For example the need to go to the store for some other reason may make the agent more willing to make the trip, lowering the effort for this alternative to an acceptable level, or the agent may recall a store that will be open despite the late hour. In the same time span, however, further constraints may be realized and others may expire, which may in turn alter the constraints on effort and compatibility themselves. For example, other tasks may be completed and the agent may be willing to spend more effort toward obtaining milk. At a more global level, waiting may lower the whole importance of making tea, lowering compatibility and allowing abandonment of the intention. Time is a natural feature in deliberation that has often been downplayed in AI planning systems. It is not until it becomes a direct consideration in the architecture itself, as it is here, that its true significance becomes so obvious.

---

86 Of course, even then the information will still never be complete, because the agent can never have complete knowledge of the world around it. From the point of view of the agent, however, consulting every concept is as complete as it can possibly get.

When effort and compatibility are used to compare alternatives to a particular intention, utility is still important. That is, these are not substitutes to utility, but go hand in hand with it (since utility also incorporates many other factors). The effort and utility of alternatives may make one choice obvious over another to be applied to a particular intention, but the utility assigned to the option that is acceptable in terms of compatibility and effort may still not be enough to make it the best choice. Some other alternative, contributing to some completely different intention may be better. For example, the agent may require boiling water for tea and have the option of using a stovetop kettle or an electric kettle. While these options are available, the telephone may ring. One or the other of the kettle options may be better in terms of effort and compatibility, but answering the phone will still take priority most of the time.

The final factor useful in deliberations, as mentioned earlier, is timeliness. Timeliness does not operate in the same fashion as utility, effort and commitment, but rather operates at an extremely local level. When a timely response is required, this fact is stored with related alternatives. This may bias the agent (depending on the effort, compatibility, and general utility of the timely alternatives) to take advantage of methods of selecting one alternative over another that do not involve waiting for more information. For example, if the agent is in the middle of putting milk back in the refrigerator and the telephone rings, this is a known time-constrained opportunity. The fact that this exists may (assuming the estimated utility is greater than what it is doing), immediately cause the agent to stop what it is doing and answer the telephone. It may even, in the case of close estimates of utility, cause the agent to be biased toward the timely alternative even when that alternative's utility is not the greatest - by virtue of the fact that its estimates will be more accurate than others, and there is no time to obtain any more information). Once again, the methods by which this occurs will be discussed in the next Section.

To summarize, the key to deliberation in this architecture is *time*. Time allows the agent to gradually consider its information, using the most obvious estimates when little time is available for making decisions, and when the situation requires, waiting and gathering more information. The distribution of deliberation over time is also what prevents the agent from being overwhelmed with background constraints when following a routine. When alternatives from some intention first become available, they will be based initially on only the most obvious knowledge the agent possesses: constraints in the intention itself (mainly preferences) and interactions

between the core of the intention and other concepts and intentions the agent already has in working memory. In the majority of cases, this is enough: the agent can commit to some (mental or physical) action and move on. However, when it is not enough (i.e. when more global constraints on utility, or the effort and compatibility associated with some intention are strong), the agent can spend more time gathering information. Much of the time however, the agent simply does not bother to go to the extent necessary to check all this information. The primary features of this architecture in terms of dealing with everyday activities are thus the structure of intentions (the connection of distributed background knowledge to the core of intentions either closely or more distant based on relevance and experience), and this approach to deliberation (taking advantage of this distribution and the natural effects of time on activity).

It is also important to emphasize that the rest of this architecture also operates over time, and the constraints that give rise to particular alternatives can, from one cycle to the next, keep proposing them. In the replacing-milk situation above, if the agent chooses to answer the telephone, the constraints that give rise to the idea of returning the milk to the refrigerator will, for the extent of their lifetime, keep proposing this as an alternative. Realistically, the telephone conversation would have to become very tedious for an agent to decide that putting milk in the refrigerator is more important, but in other situations this does occur: if the phone conversation comes while water is heating, and boiling water is left on the stove, for example.

5.4. Constraint-Directed Control

Thus far, two contrasting roles played by constraints have been emphasized in this Chapter. Sections 5.3.1 and 5.3.2 have emphasized the guiding role of constraints. In this role, I have illustrated the use of constraints in contributing and removing utility from alternatives for action, directly guiding decision-making in activity. I have also shown that constraints can apply at a higher level, both through inheritance and by directly modifying the alternatives an intention provides. Section 5.3.4, on the other hand, has emphasized the more global nature of control constraints: in particular, how global constraints on utility and intention-specific constraints on effort and compatibility can allow the agent to make rational decisions.

However, the role of constraints in control in this architecture goes much further than this. As illustrated in Figure 5-14, constraints can also directly

control deliberation, recollection, and perception, the three primary processes of this architecture.

In perception, expectation constraints (representing the expected results of pending actions) form a particular focus for the perceptual component, allowing it to distinguish particular items that are useful in the agent's current activity. When picking up a cup, for example, an expectation of the cup being in the agent's hand would be posted, and could be confirmed or refuted by perception.[87] *Focus* constraints can also be triggered, directly forcing perception to focus on particular objects. Such constraints are essentially a command to perception that overrides all other options. Once again, each constraint will have a utility, and this utility allows the agent to deal with the most important constraints first.

Constraints can also be used as a mechanism for appropriately adopting memory management policies. As mentioned in Section 5.3.3, several options exist for memory management: as new items flow into working memory, older, less important, or less frequently used items can be removed. A particular constraint, triggered by external circumstances that indicate a particular strategy should be used, can move the agent from one strategy to another. For example, if the agent is quickly switching from task to task, high level constraints can recognize this and cause the agent to remove less frequently used items as opposed to using the other approaches. Like perception, explicit focus constraints can also be applied here. A constraint may be triggered in a particular situation (e.g. when one intention is far more important than any other), indicating that memory management should focus on one particular intention and its related concepts to the exclusion of others. Such a constraint could be used in important situations to direct the entire architecture toward one particular activity: preserving one intention's detailed concepts over others would allow more constraints to contribute toward alternatives for one particular intended activity. This would not exclude other activities completely: a constraint of high utility could still contribute some other alternative that would be significant. This merely

---

87 In models of intelligent agents, perception is usually confined to the visual: the agent would have to see the cup. However, there is no reason that other modes of perception could not also be controlled in the same manner. A posted expectation for a cup being in the agent's hand could be verified by touch for example, as well as sight. Similarly, the sound of a cup being placed on a tabletop could confirm this expectation.

allows the agent to focus on one particular activity and put others further into the background.

In deliberation, we have already seen that a constraint on utility is necessary. This is another constraint that directly controls the agent's activity, by being modified when significant situations are recognized. For example, a *be careful* constraint, in addition to putting forward particular alternatives for action, and modifying an overall intention, may direct the agent's utility threshold upward or downward. Effort and compatibility constraints have similar controlling effects: global constraints can also adjust levels of effort and compatibility defined for a particular intention. The *be careful* constraint mentioned above, for example could require extra compatibility. A social setting would alter compatibility in a similar fashion: we do things better for company than we do for ourselves.

In addition to these examples of control constraints, Section 5.3.4 has also mentioned that there are many constraints that recognize when more information for a particular alternative action should be found before committing to it. These constraints also indirectly control deliberation, in that they force a delay before choosing a particular alternative and allow additional information to be considered for all possibilities. Such a delay may allow some alternatives to become better (higher utility), others to become less appropriate choices (greater effort, less compatibility, lower utility), entirely new choices to arise, and other (time-dependent) opportunities to be missed.

While some of these high-level control constraints may be attached to concepts or intentions (e.g. constraints that recognize when further deliberation is necessary in regard to a particular intention), many are not. It is difficult to see how a global utility constraint, for example, could be attached in such a manner, or how constraints that recognize interactions between intentions (e.g. a focus constraint for memory management that recognizes when one activity is far more important than others). These constraints must be viewed as constraints on all activity in which the agent can participate, unattached to any particular intention, and permanently available in working memory.

Because of their generality, there will not be many of these constraints: they do not contribute toward any particular activity, but modify the internal components of an agent in order to allow it to better make decisions on actions as the situation around it changes dynamically. They are the behaviour level constraints described in Section 5.1.3.

These constraints are significant in this capacity, in that they represent knowledge outside of particular intentions and concepts in the agent's working memory. However, in comparison to these, their role is minor. As emphasized throughout this Section, the most significant part of this architecture is the appropriate recall of concepts and intentions into working memory, and the subsequent contribution of constraints that modify intentions and directly propose or discourage potential actions. These behaviour-level constraints do not perfectly direct an agent in every new situation. This must be accomplished though the application of intentions and the background knowledge behind them, as detailed throughout this Chapter. *Rather, these constraints provide defaults for the processes within an agent that allow it to apply constraints and background knowledge effectively.*

While individual control constraints will often temporarily override the initial defaults provided, it must also be recognized that there will be certain specific situations in which these control constraints will interact to alter the behaviour of all components within the agent. For example, when a strong time-limited opportunity is available, such as answering the telephone, a lower utility, stricter effort limitations for other pending intentions, and perceptual focus constraints may be automatically triggered. Now, these may be triggered individually through the recognition of the significance of the phone-answering alternative in relation to others the agent is currently considering. However, this may also be handled in an activity-independent fashion. The recognition of a time-dependent opportunity may perform such control functions automatically, over and above any specific knowledge of answering the telephone. A package of control constraints such as this allows the agent to temporarily override its current settings and adopt a different *demeanor* toward activity. Within the context of this architecture I term such a package a *behaviour mode* .

At first, this may seem an overly complex mechanism for controlling behaviour, with far too many potential behaviours to mediate between in order to keep activity rational. Indeed, in Section 3.3, I criticized Brooks' view of activity as competing and layered behaviours for precisely these reasons. However, the behaviour modes proposed here bear little resemblance to what Brooks proposes. In Brooks' work, behaviours represent and propose specific actions for agents to take; behaviours form a layered architecture, control moves from behaviour to behaviour, and rational activity emerges. Behaviour modes as I have proposed them are *not* behaviours in the sense that Brooks proposes. They are groups of a few general control constraints

that allow the agent to tune its internal components toward exceptional external situations. Their sole purpose is to provide a mode of control: they generate no alternatives for activity, recall no information, and make no direct decisions. Because of this role, and the nature of everyday activities, an agent will have only a few behaviour modes. Everyday activities are not complex, and there are only a very few basic exceptions to the agent's default mode of control (e.g. the time-dependent opportunity example described above). While there will often be situations in which an individual constraint will slightly alter the behaviour of one or more components (e.g. a circumstance indicating that some alternative deserves further consideration during deliberation), only rarely will large-scale control modifications on the order of a behaviour mode take place. I have identified only especially timely situations, extreme danger, and emotional reactions such as fear. Within the context of everyday activities, the former (and the example used throughout this Section) is really the only behaviour mode that applies. I will not argue that this is the only behaviour mode that is applicable to everyday activities: I will argue that these satisfice for the majority of everyday activities, and future research will address the role of behaviour modes in this architecture.

5.5. Summary

This Chapter has presented a constraint-directed architecture for improvisation based on the requirements of everyday activities described in Chapter 2 and the nature of improvisation described in Chapter 4. I have described constraint-directed knowledge representation structures that capture both the routines an agent applies during the course of everyday activities, and the background knowledge that allows those routines to vary with the situation in which the agent finds itself. I have also illustrated the interacting processes involved in applying these structures to perform everyday activities, and have described in detail the memory, perception, and deliberation components that make up this architecture.

Within this model, constraints are applied at several levels. At the lowest level, individual constraints in the agent's working memory directly support or refute alternatives for activity. Constraints also directly modify intentions, and at the highest level, directly control the processes of the architecture itself. Overall, constraints bring to this architecture precisely that which is needed to deal with everyday activities. Constraints provide a mechanism for adding flexibility to the agent's routine knowledge. Constraints also directly represent the known restrictions on activity dictated by the environment around the agent, the restrictions indicated by adopting particular choices of

action, and the interactions between activities the agent is considering. That is, they provide an useful mechanism for exploiting regularities in the world around the agent and in its own knowledge of activity. Constraints have also been shown to be useful in controlling the extent to which an agent reasons during the course of an activity, and also in providing a flexible means of control over the processes in the architecture itself.

Constraints have been recognized previously as a useful mechanism for providing flexible and adaptable control. In particular, Evans and Anderson [1990] have presented a means of control in multi-agent planning that is similar to the utility-based deliberation methods described in Section 5.3.4 The similarities lie in the extent that constraints contribute to the deliberation process and in the use of a threshold for utility. The applications themselves are very different, however, and the constraint-directed deliberation described in this Chapter has many novel aspects. These will be discussed in Section 8.5.

While this Chapter has provided an extensive description of this architecture and its internal operation, many implementational details remain: the nature of memory, for example, and whether a qualitative or quantitative representation of utility should be used, have been left to be decided upon in a particular implementation. Chapter 7 discusses alternatives for these choices, and describes a partial implementation of this architecture, including several examples of improvised behaviour. The following Chapter describes the tool created as part of this research for the implementation of an improvising agent.

CHAPTER 6

A GENERIC SIMULATION SYSTEM FOR INTELLIGENT AGENT DESIGNS

The universe is wider than our views of it

– Henry David Thoreau, *Walden*

6.0. Introduction

In order to examine the behaviour of an intelligent agent architecture, that architecture must be given a physical embodiment: the agent must be allowed to interact with an encompassing environment and display its behaviour in light of stimuli from that environment. Such an embodiment is required during the development of an architecture, for the purposes of feedback as well as for the purposes of testing the final design to illustrate both the competency and the limits of an architecture. For the purposes of this research, it was decided very early that a simulation system would provide the most appropriate environment for development, for reasons that will be made clear in this Chapter.

A simulation system for intelligent designs requires special elements not associated with general-purpose simulation languages and equivalent tools. For example, in order to be used for such purposes, a simulator must be generic: that is, it must provide the flexibility to accommodate an architecture that is undergoing constant modification, as well as the ability to represent a wide range of environments. The lack of such flexible support in existing simulation systems led to the development of *Gensim*, a simulation system designed in conjunction with the architecture described in the previous Chapter. Gensim functions as a generic simulation system, and was designed not only to support this architecture during development and offer facilities for examining its performance, but to provide the facilities necessary to be useful to a wide range of intelligent agent research.

This Chapter describes the motivations behind the use of simulation in intelligent systems development, both in general and with regard to this architecture. These motivations are then used to define a set of criteria for a generic simulation system: one that will support a wide range of agent designs and a wide variety of environments. These criteria serve two purposes: as a benchmark by which to compare existing systems, and as a set of goals for the development of Gensim. This Chapter then describes the features and functioning of Gensim in detail.

6.1. Simulation and Intelligent Systems Research

Intelligent agents designed to perform in the real world should by definition be tested and evaluated in the real world. However, intelligent systems continue to be developed using simulated worlds despite this obvious fact: typically, an agent is designed to demonstrate some aspect of intelligence and is tested in simulation, with the assumption that the demonstrated behaviour will scale appropriately to the real world. However, the continuing disparity between the relatively small number of deployed systems and the large number of systems that perform only in simulated environments has come to be one of the strongest and most often cited arguments against the utility and future of intelligent systems (e.g. [Dreyfus, 1981; McDermott, 1981]). This criticism is not undeserved: in many early cases, simple simulated worlds were chosen innocently, under the mistaken assumption that the differences between such environments and the real world were insignificant. However, many systems are tested in simulated environments that make unreasonable simplifying assumptions in order to avoid issues that would be problematic to the system. Very few of these are deliberate attempts to mislead, but the poor science demonstrated in the unrealistic

assumptions made by such systems has led to suspicion of any modern intelligent system that performs in a simulated environment.

The difficulty with this suspicion is that it confuses problems that may arise from using a *simplified* environment with those that arise from using a *simulated* environment. *Simplified* environments can never be like the real world, and only intensive analysis of the assumptions made in constructing these worlds can gauge the applicability of the results obtained from them. A simulated environment need not be an overly simplified model of reality. Indeed, even simplification is not in and of itself a problem. Pollack (in [Hanks et al., 1993]) goes to great pains to illustrate that not only is simplification not a problem, it is as absolutely necessary for experimentation in AI as it is in any science. Simplification is required to isolate a given phenomena in a complex system, and is a critical tool to any scientific endeavour: any difficulty lies in the lack of analysis of assumptions made when using a simplified world. Rather than qualifying results obtained from such research, general (and sometimes grandiose) claims are often made that are not logically entailed by the research itself.

When (in AI) we state that the environment a given agent operates in is a *simulated* one, it means only that the objects around the agent and the physics of the world itself exist only in the form of a computer program. The agent interacts with information provided by the program rather than the physical objects themselves. This program can be as complex or as simple as desired: simulation does not necessarily imply over-simplification. The suspicion of simulated environments is justified in that in many cases, simplification is hidden beneath simulation. In any truly scientific effort, however, the two issues should be dealt with separately.

Despite this overall suspicion, there are many logical reasons for using simulated environments for testing intelligent agents. The foremost of these concerns the nature of the field of artificial intelligence itself. As discussed in Chapter 1, AI is an immature science, and as such it does not yet possess a wealth of broadly accepted theories that form a concrete foundation for ongoing research. Because of this, any broad research will be hampered by being forced to deal with unsolved problems in peripheral areas. Nowhere is this difficulty felt more than research involving intelligent agency. Almost any research project in this area involves the integration of a number of areas (e.g. vision, knowledge representation, robotics), with the necessity of solving open problems in those areas or making assumptions about the nature of those problems. Thus, if one develops a theoretical architecture for an intelligent agent, unless one intends to solve *every* open problem in areas

such as computer vision and robotics, one is forced to make (possibly incorrect) assumptions about how these various peripheral aspects will operate when more complete theories are available.

The use of a simulator in intelligent agent research is thus largely an issue of practicality: the large collection of interrelated problems encountered when implementing an intelligent agent necessitates the use of a simulator to aid in accounting for those pieces of the theory that are not yet complete. The focus of any individual research project in intelligent agency may be quite narrow (e.g. demonstrating the utility of a new form of representation within an intelligent agent). In such cases, it is irrelevant (from the point of view of the original research) to provide complete answers to peripheral problems such as sensing, given the effort required to solve these problems. Admittedly, peripheral areas such as sensing are not independent of other aspects of intelligent agency, but can be de-emphasized as long as the assumptions made to deal with them are explicit and reasonable, and that results from such research are interpreted in light of those assumptions. Indeed, Pollack (in [Hanks et al., 1993]) points out that the assumptions made about peripheral areas can suggest further experiments: an important consideration given AI's current stage of maturity. In addition to conserving research resources, simulators are also often required because of a lack of physical resources. Few research facilities can afford the staggering cost of supplying enough robotic technology to meet the needs of all the intelligent systems research projects they support [Etzioni and Segal, 1992] nor the manpower to maintain this equipment.

In addition to basic limitations of research resources, there are many sound reasons for choosing to use a simulator to examine the performance of intelligent agents. These reasons essentially fall into two large categories: some advantages come about because of the utility of using software rather than hardware, while others arise from greater control available in a simulated world compared to the real one. Some of these have been noted in previous studies (e.g. [Cohen et al., 1990; Etzioni and Segal, 1992; Etzioni, 1993]), while others have arisen during the design of the architecture presented in the previous Chapter:

- *Maintaining control and isolation.* The real world is often unpredictable, and it can be difficult to establish control over those variables that must be controlled experimentally and isolate those that are to remain independent. Indeed, in the real world, it is impossible to control for many variables. Just as other scientific experiments are performed in specialized laboratories to isolate the particular phenomena the

experiment is concerned with, simulators provide the control necessary to extract valid experimental results. Controls may be placed to eliminate interference, but also to provide control over the complexity of the domain, in order to find the point at which techniques undergoing experimentation begin to break down. Because of these control abilities, domains can be saved and shared, allowing simulation environments to serve as a broad metric for comparing intelligent agents [Howe, 1993; Hanks et al., 1993].

- *Maintaining consistency between trials.* This is another aspect of control: in the physical world, it is impossible to guarantee exactly the same conditions between trials of an experiment. A simulator can present a common environment across many trials, providing exactly the same initial state and planned course of events for each. This may be extremely difficult or impossible in a physical environment.

- *Ease of experimentation.* A simulator can also perform a long series of trials with much greater speed than could ever be expected of a physical environment. Once constructed, simulated domains can also be modified, either slightly or in the extreme, much more quickly than can most physical domains.

- *Evaluating agents in inaccessible worlds.* Many of the worlds that intelligent agents are designed to operate in may be physically inaccessible for the purposes of testing (e.g. space probes) or may be worlds that do not physically exist. In such cases, simulation is the only viable option.

- *Creating worlds more demanding than the physical world.* While the physical world is already more complex than current technology for intelligent agents can handle, we may still wish to test agents in domains where some aspect of the physical world is amplified to an extreme. For example, if a given event occurs rarely in the physical world, but it is critical that an agent handle such an event correctly, it is much more convenient to simulate the event than to wait for its occurrence in the physical world.

- *Examining agent performance in dangerous environments.* In many cases, intelligent agents are expected to perform in domains that may be too dangerous to humans. In these cases, the domain is presumably dangerous for the intelligent agent itself, and thus it is worthwhile to

verify its correct performance in simulation before risking valuable software and hardware in such an environment.

- *Avoiding component unreliability.* Current hardware technology causes many problems in demonstrating AI systems. For example, robotic units often break down and cause errors unrelated to the intelligent system under evaluation. Such failures can sometimes occur with great frequency [Etzioni and Segal, 1992], greatly slowing down the research process. Given that most current research will be near or beyond the limit of current technology to adequately support, it will often be more reliable to examine the behaviour of the system under simulation rather than in the physical world. Indeed, many laboratories with a wealth of robotic equipment continue to use simulation for precisely these reasons. Etzioni and Segal [1992] also point out that current robotic technology can also severely limit realism: manipulators and visual sensors often cannot perform to the degree that is required in some domains. This necessitates the same kinds of simplifying assumptions for which simulators are criticized. Indeed, many projects involving the use of robotics artificially manipulate and simplify the physical domains in which they operate to cope with hardware limitations [Etzioni, 1993].

- *Maintaining a consistent agent interface.* Depending upon the design of the simulator, it may be able to provide an identical interface between agents (e.g. providing an identical basic interpretation of sensory information to each agent). This allows an agent's reasoning abilities to be examined independently of its sensory abilities, and for agents to be compared solely on this basis. In many cases, this is the primary reason for using a simulator: sensory processes often require special equipment and significant effort to realize, and are complex enough that a comparison of two different agents could be significantly biased due to differences in sensory processes.

- *Providing an understanding of domain complexity.* By allowing control of specific aspects of the domain, a simulator fosters a deeper understanding of the domain, and can also facilitate more specific analysis of domain complexity. Some of the most significant results of simulated systems such as Phoenix [Cohen et al., 1989] have been a better understanding of the complexity and nature of the environment through the process of implementing a simulation.

This is not to say that simulation is always an adequate substitute for reality: the use of a simulator for evaluating intelligent agents is by no means without disadvantages. Howe [1993] points out that the very ease of use that simulators bring to the testing process can allow researchers to construct experiments too quickly, without the clearly stated hypotheses and careful methodology that physical domains encourage. This problem has been shown to be especially prevalent in military modelling, and has come to be known as the *base of sand* problem [Davis and Blumenthal, 1991][88]. However, the most obvious difficulty with employing a simulator has already been alluded to: the ability to make simplifying assumptions (even to avoid issues that are impossible to deal with otherwise) can easily lead to making broader, invalid assumptions about the way the world works [Agre, 1988]. This problem has also been noted in other fields where simulation is employed: In general, the control that simulation models provide can also allow researchers to model only those aspects of an environment that they choose to be significant, ignoring other aspects that may have a much greater impact on results than is initially assumed [Friedland, 1977].

In the past, such assumptions have led to an overall lack of understanding of the physical world and how agents operate in it (which in turn has resulted in many of the limitations of classical planning theories). However, relying on a completely-implemented peripheral system, as many recent systems attempt to do forces much of the overall research effort to be expended on intricacies that are not the real focus. It also forces the reasoning in such systems to operate at a low level, and deal with simple environments: the focus is on making complete simple systems rather than complex systems that do not attempt to provide an answer for every question. Systems that include complete sensory and effectory apparatus (e.g. [Agre and Chapman, 1987; Chapman, 1990; Maes and Brooks, 1990] have illustrated that such approaches will work in highly reactive domains where few if any high-level, long-term decisions are required. However, more complex domains are beyond the scope of these approaches. This is the real difficulty: such models are becoming more attractive because of their simplicity, and there is a growing and unjustified assumption that these same models will suffice for the

---

88 Davis and Blumenthal [1991] explore many of the factors behind modelling experiments in military applications in particular that give rise to the problem alluded to by Howe. These factors, such as dubious acceptance criteria, minimal empiricism, and parochialism, occur much more generally outside of military applications than Davis and Blumenthal state.

implementation of higher-level behaviours[89]. Clearly, both approaches that focus on the complete implementation of simple systems as well as those that focus on given aspects of complex systems have their roles to play in intelligent systems research. Both perspectives approach the same puzzle from opposite ends, and each makes its own assumptions. What is important from a scientific viewpoint is that those assumptions be explicit and that results produced be explained in light of those assumptions.

The decision to make use of a simulator for the development of the Waffler architecture was due to a combination of these reasons. By its very nature, the architecture cannot provide answers for all questions: it is an illustration of an approach to a particular cross-section of human problem-solving, and was not designed to answer every question regarding the lower level cognitive support for those questions: designing the sensing aspects alone would be a large, long-term research project. Not answering all these questions necessitates the use of a simulator. Attempting to employ robotic technology would have restrained the architecture to very simple domains: an agent could not move about a kitchen, finding objects such as tea bags and reasoning about whether to put milk away, simply because this level is beyond what current hardware can realistically support. Such a move would invalidate the architecture itself: improvisation is an approach designed for complex but well-experienced domains. Trying to illustrate it in a simpler domain is simply not demonstrating improvisation. Most of the other factors were also involved: robotic equipment was lacking, for example, and there was from the beginning a strong desire to have an easily varied, controllable domain.

---

89 Indeed, the premise of much of this work is that the various behaviours of simple animals can be understood and implemented much more easily than human behaviours, and that these mechanisms can serve as a basis through which higher-level intelligence can be realized. In a weak sense, I believe this is true: the implementation of such behaviours can illustrate interactions and techniques which may be useful in creating more complex behaviours. However, the strong sense in which Brooks and others imply this is unreasonable: the behaviour of lower animals is simply too far away from human-level intelligence to be a convincing basis. Many believe that the evolutionary process itself makes it futile to try to find any similarity in cognitive machinery between different animal species. Different species will differ radically in the nature of their intellectual processes, simply because of differences in their environments and the expectations this places on cognitive mechanisms [Pearce, 1987]. Moreover, even the behaviour of lower animals employs many problem-solving mechanisms that are beyond the scope of these simple architectures [Pearce, 1987; Birch, 1945].

6.2. Requirements of a Simulation System for Intelligent Agents

Having made the decision to employ simulation technology, further questions arise as to the role that technology is expected to play and the specific features of that technology that are required. Simulation systems have become an important tool for examining real systems under research conditions, in fields ranging from physiology to natural resource management. A simulator provides a model of a real system, allowing the modeller to study how that real system behaves under conditions of interest, and examining the consequences on the entire system of a change in some aspect. The primary function of a simulation system is thus to manage change in a model over the course of time. In many areas, the overall objective view of the environment is of prime importance, and various sources of change in the environment need not be differentiated. Such is the case in most simulation for natural resource management, for example: the focus of interest rests along the whole scope of the environment.

Change in a such a simulated world comes from three sources: from random elements in the environment (e.g., the wind blowing), from objects that perform mechanically and deterministically (e.g. machines), and from intelligent agents, which make changes through actions they take of their own volition. For the purposes of this discussion, the latter type will be termed *primary agents* or simply *agents*, and the former types *random* and *controlled* agents respectively. Change from from random and controlled agents is supported easily in most simulation environments: even older simulation languages such as SIMULA [Birtwhistle et al., 1973] provide the features necessary to support mechanistic change. Supporting truly intelligent agents, however, requires many additional facilities from a simulation system. In addition to providing an accurate model of a particular environment for the user of the simulator, a simulation system for intelligent agent designs must provide a virtual reality for the agents existing in the simulated environment. It must provide a view of the world to intelligent agents through their own perceptual systems, and integrate the actions of these these agents (through their effectory systems) with change from other sources in the simulated environment [Anderson and Evans, 1994].

In a simulation system for intelligent agents, neither of these two roles can be emphasized at the expense of the other. The current stigma attached to using simulation systems in conjunction with intelligent systems development illustrates what happens when sacrifices in the realism of a simulated environment are made. Similarly, if the accuracy of the environment

surrounding a collection of intelligent agents is stressed over accurate interaction between the agents and that environment, the performance of those agents will not reflect reality.

The ability to provide a virtual reality for intelligent agents is a broad goal of simulation testbeds for intelligent agent development. Any individual research project will have many more specific needs, however. We may desire to examine a single agent design across a variety of problem domains, or to examine the performance of a range of agents within a single domain. The particular focus of each research project dictates additional requirements: one may be interested in decision-making in a single agent, for example, or cooperative behaviour between multiple agents. To deal with such a range of interests, any simulation system designed for intelligent agent testing must be *generic*: that is, it should make as few assumptions as possible about the nature of the environments it is modelling and the agents that inhabit those environments. It is impossible to completely eliminate all such assumptions, simply because agents and the simulator must communicate with one another in order to provide perceptual information to agents and to inform the simulator of the agent's actions. Such communication mechanisms constrain the design of agents and the design of the simulator itself. However, like all other assumptions, their impact should be minimized. In computer science, there is usually some compromise that must be made when designing a generic system: the more generic the system, the wider the range of applications for that system; however, this generic nature is usually accompanied by an increase in complexity (e.g. the number of options a user is confronted with), making the system more difficult to make use of[90]. This is equally true for simulation systems. A generic simulator will be more widely applicable than a special purpose system, but is likely to require more work on the part of the user to employ, since fewer assumptions about the nature of these entities are made by the system.

In addition to placing as few constraints as possible on potential agents and environments, a generic simulator must endeavour to provide the features necessary to support the wide range of projects in intelligent agent research described above. In the design of the architecture presented in the previous

---

90 Compare, for example, the TEX and Microsoft WORD text processors, or EMACS and a standard Macintosh text editor. All are useful, but the more complex systems, while having a steeper learning curve and expectations of greater knowledge and work on the part of the user, are applicable to a wider range of applications.

Chapter, as well as other research [Evans et al., 1992], the following requirements were noted:

- *Modularity.* A simulator should ideally be a completely separate computational process from the agent and the domain it resides in. This would result in a truly general simulator in that one could theoretically plug any agent or domain definition into it.

- *Support for multiple agents and processes.* A simulator that supports multiple agents is clearly more widely applicable than one that is restricted to single-agent capabilities. Indeed, few realistic domains are exclusively single-agent domains, and one common criticism of an intelligent agent design is a lack of provision for reasoning about other agents. Thus, the ability to support multi-agent simulations is crucial for a generic simulator. Even though much of the coordination abilities in a multi-agent environment must reside in the agents themselves (e.g. reasoning about how other agents may interfere with one's actions, preventing and correcting for such interference), special support for multiple agents must be implemented in a simulator as well. This includes aspects such as separate storage space for the knowledge of multiple agents and the ability to handle change from several different agents at the same time.

  To be able to handle the widest possible range of abilities, a generic simulator should also be able to work with agents that consist of multiple processes, as well as single-process agents. A lack of support for multiple processes in a simulator would exclude parallelism from any agent. Parallel architectures are being turned to increasingly to deal with the large volume of information inherent in most problems involving AI, and so simulator support for such architectures is of increasing importance.

- *A clearly defined relationship between agent and simulator.* Given that a generic simulator is to be used to run an arbitrary collection of agents in a given domain, the relationship between agents and their simulated environment must also be clear and explicit in order to ensure the accuracy of the simulation. Among other things, any assumptions about the timing of actions, chains of causation over time, sensory abilities, and way in agents communicate (e.g. refer to objects) with the simulator must be clearly understood by modellers to ensure accuracy and utility.

A lack of precise definition of such features has traditionally been a problem in simulation systems. Details of communication between an agent and simulator are often sketchy, as are assumptions made by the simulator about the internal operations of agents and other parts of the domain. Indeed, basic concepts such as what the simulator considers an "action" or an "event" to be is often left to speculation. This makes it exceedingly difficult to assess the suitability of a simulator for a given application.

- *Simple interface.* The interface between the agent and the environment should be as simple and low-level as possible, to reflect the real world (and rely on fewer assumptions about the domain and agents). The input of the simulator (output of primary agent processes) should ideally be the direct actions that the agent is performing on each object in the environment (e.g. grasp teapot; lift teapot). The agent will have its own expected set of results for each action it is carrying out, and the simulated environment confirms or refutes these expectations and carries on a simulation of a causal consequences of the actions of the agent. For example, the agent may pick up the teapot and expect the teapot to then be in its hand. The simulator, upon receiving the information that the agent is carrying out this action, can then update the environment accordingly: depending on the physics and other conditions in the environment, the teapot may or may not actually wind up in the agent's hand. The agent can then be informed of what actually occurred.

- *Ability to control for many variables.* A generic simulator, in addition to being able to handle a diverse array of agents and domain definitions, should be able to control for many aspects of the functioning of the world. The actual domain features that must be controlled vary from experiment to experiment, but a general simulator should provide for control of elements that are common to many domains (e.g. how much sensory information is presented to the agent, how often the agent receives such information, how often the environment may be altered). There are two basic types of variability that can be supported by a simulator. Examples such as those just cited can all be implemented using simple parameters to the system: a global parameter for the length of time between environmental updates, for example. On the other hand, there are types of variability that are too complex to be handled by a simple parameter mechanism. For example, two agents may work with actions at radically different degrees of granularity. One agent may reason about actions like *lift* to grasp an object, while another may intricately control the

movements of a manipulator to accomplish the same thing. Granular variability can still be supported by a simulator, but is clearly much more complex to change.

These criteria represent the basic requirements of a generic system for testing and examining designs for intelligent agents. While they were arrived at through the analysis of the the needs of the Waffler architecture (Chapter 5) as well as other ongoing projects in intelligent agent research at the University of Manitoba (e.g. further work on constraint-directed multi-agent planning [Evans et al., 1992], I believe these projects to be representative of current research trends in intelligent agency, both in terms of their general objectives and their requirements for simulated environments. Other suggestions for requirements exist: for example, [Hanks et al., 1993] review specific issues in intelligent agent research, with the implication that simulation must look toward providing support for such issues. Many of these points are subsumed by the very fact that such a simulator must be as generic as possible. For example, supporting a wide range of environments entails several of the requirements of Hanks et al. [1993], such as providing for external events and using a well-defined model of time.

6.3. Evaluating existing simulators

In recent years, several simulation testbeds have been developed for the purpose of testing and examining intelligent agent designs. These include: *Phoenix* [Cohen et al., 1989; Howe and Cohen, 1990], a generic simulator that is used as part of a larger system for controlling fire-fighting agents; *Tileworld* [Pollack and Ringuette, 1990], a grid-based system for examining the performance of intelligent agents in domains where the stability and importance of goals varies; *Mice* [Durfee and Montgomery, 1989; Montgomery et al., 1992], a multi-agent testbed also based on a grid-like structure; and *Ars Magna* [Engelson and Bertani, 1992], a sophisticated system that attempts to accurately simulate the functionality available in current robotic technology. While each of these systems is powerful and sophisticated, each is also geared toward specific environments and agent designs. Each of these simulators was examined in light of the criteria presented above, in order to gauge their ability to be used in this project. This Section summarizes the results of this analysis.

6.3.1. Phoenix

Phoenix [Cohen et al., 1989; Howe and Cohen, 1990] is best known as an advanced planning system for fighting forest fires, incorporating both reaction and deliberative planning. In addition to an advanced planner, however, Phoenix also contains an integrated simulation system for Phoenix agents. This subsystem accurately simulates the spread of forest fires and the activities of agents attempting to extinguish these fires. The Phoenix simulator manages a great deal of unpredictability in its domain: actions can have random components, and provision is made for random agents (e.g. fires). The domain provided in Phoenix is extremely realistic, and the overall system is impressive. The simulation component of Phoenix timeshares the agents and environmental updates as a single operating system process. Synchronization is set by default at five minutes (that is, the system works to maintain each agent and the environment to within five minutes of one another); however, this synchronization "window" can be made larger or smaller, and run time decreases or increases in proportion.

While its success as an overall system is unquestionable, as a generic simulator Phoenix leaves much to be desired. The authors of Phoenix claim that the system is generic, in that it can manage any simulation that involves maps and processes [Cohen et al., 1989]. However, this generic nature is extremely limited in practice. In the sense that the map provided for the system could be a map of a city as opposed to a park, or that the agents defined could be cars as opposed to bulldozers, the system is indeed somewhat generic. However, there would seem to be significant difficulties involved in extending the system to domains without very strong analogies to its fire-fighting domain (no attempt has been made to adapt the simulator to another environment has been published to date). The simulator itself is a secondary issue in the Phoenix project, and is not well described. No information is given on the interface between the agent and simulator. No algorithmic descriptions are available apart from the basic claims of how timing works, and even those timing examples are not complete: there are no illustrations or examples of how agents are interleaved with one another, nor how the interactions between them are handled at a low level. The examples cited [Cohen et al., 1989] show processing for only one agent and do not give any indication as to how other agents are treated.

Phoenix is also claimed to be a multi-agent simulator, and in some sense this is true: multiple agents can inhabit the same domain. However, only one agent is truly a primary agent: the *fireboss*, a centralized controller that issues instructions that other agents carry out. These alternate agents more closely

resemble what I have defined as controlled agents, in that they are largely under control of the fireboss. They differ slightly from controlled agents, however, in that they have their own simulus-response mechanisms that operate independently of the fireboss. They also have their own views provided by the simulator, and these can differ from the objective view maintained by the simulator[91]. It can be argued that this is simply a very centralized organization of a multi-agent system, similar to a blackboard organization [Nii, 1989; Hayes-Roth, 1985]. However, the limited nature of all but the central agent in any demonstration limits the utility of the simulator's multi-agent capabilities. While all Phoenix agents share the same basic architecture [Cohen et al., 1989], it is not clear how much difference in architecture can be tolerated by the basic system, nor how adept the simulator is at handling multiple primary agents in general.

Phoenix also has only a very limited number of parameters that can be adjusted. The only one mentioned in the literature is the time difference that is allowed between the environment and the agents in the system [Cohen et al., 1989]. All others (presumably) must then be altered or controlled through re-implementation of the agents or environment. However, since neither the interface nor the low-level structure of these are explained, the complexity of such re-implementation cannot be estimated.

Overall, the simulation component for Phoenix, while clearly functioning very well within the sophisticated domain for which it was originally designed, is of dubious use as a generic simulator. Phoenix is representative of a class of modern intelligent systems, where a simulator (or other sub-component) is designed for a fairly specific type of agent and environment, and is claimed in retrospect to be generic. Just as some highly-successful early expert systems were presented as general-purpose problem-solving shells (e.g. *MYCIN* [Davis et al., 1977] and *EMYCIN* [van Melle et al., 1984]), such systems usually prove useful only in domains that closely match the applications for which they were originally designed. Like Phoenix, most of these systems do not provide any explicit reprogramming facilities to extend the shell to fit problems that differ significantly from the original application, and thus their use is extremely limited.

---

91 Although it is not explained how limited perception is performed, and this limited perception does not seem to be easily adjustable.

6.3.2. Tileworld

Tileworld [Pollack and Ringuette, 1990] is another system presented as a general-purpose simulator for intelligent agent architectures. Unlike Phoenix, Tileworld is presented foremost as an independent simulator, rather than as a peripheral component of a specific agent architecture.

The Tileworld simulator consists of a grid whose squares can be traversed by an agent. Some of the squares in the grid are designated as obstacles that bar passage of the agent. Some squares also contain tiles that can be shifted by the agent, while other squares contain "holes" that can be filled with a given number of tiles. Each hole has a given point value and exists for a given amount of time. The ultimate aim of the agent is to maximize a point score by filling as many valuable holes with tiles as possible in a given period of time. Tileworld provides control of many variables within this scheme: for example, the lifetime and frequency of each type of object; the distribution of scores for holes; and a possible decrease in value for a hole over time.

In combination, these variations can allow testing of an agent in many domains, and there are advantages to the Tileworld approach. These parameters provide a great deal of control over one basic theme, yielding a good testbed for experimenting with problem-solving methods. The domain is also simple enough that many important issues in the relationship between the agent and simulator simply do not appear. Issues such as delayed effects of actions, for example, can effectively be ignored. The complexity of the domain can also be studied, and the domain itself is well-understood. Tileworld has also been used outside its original development group for the testing of problem-solving algorithms [Kinney and Georgeff, 1991].

Part of Tileworld's advantage, however, is also its biggest difficulty. Only one basic domain is provided, and this seems closely tied to the design of the simulator itself. Thus one can experiment with aspects such as the timeliness of a domain, but only within the general tile-shifting environment. This environment suffices for testing basic problem-solving strategies, but its complexity is lacking for testing sophisticated agent architectures. The complexity of the tile-shifting domain certainly does not come close to the Phoenix domain. Part of the desire for a well-understood domain such as tile-shifting may stem from avoiding assumptions about perception. However, avoiding the issue completely can be as problematic as making ill-informed assumptions. In the case of Tileworld, for example, it is impossible to provide domain controls for such critical features as the information-

gathering abilities of agents, because the basic domain is too simple to require any.

Tileworld is also only a single-agent simulator. While it may be possible to modify the simulator to provide such a capability, the overall timing process and the structure of agents themselves are not well-described, and so the difficulty of making these modifications is unknown. The interface between the simulator and the single agent it operates with is also not described in detail. While the simulator is demonstrated using a specific agent design, no indication is given of the ease of using a different agent architecture with the system. However, the experiments conducted by Kinney and Georgeff [1991] mentioned above describe the use of a simplified Tileworld to examine a PRS [Georgeff and Lansky, 1987] agent, a planning agent that reacts by allowing changes in the world to alter the order in which its plans are executed. This agent does show similarities to the agent for which the Tileworld was designed, in that it uses metareasoning strategies to attempt to introduce spontaneity to traditional means-end reasoning. Indeed, Kinney and Georgeff draw many analogies between these systems. However, there is also enough difference between the structures to show that the simulator can be adequately used with other agents. Kinney and Georgeff [1991] do not discuss whether extensive reprogramming of the simulator was required in order to make use of this agent, nor how much effort was needed to modify the Tileworld to suit their experiments. In addition to these limitations, many further examples are described by Hanks et al. [1993].

Tileworld, then, represents the opposite end of a spectrum of simulators. It is relatively simple to use and provides a constrained environment for ease of experimentation. However, it is limited in that it concentrates on providing a limited number of controls over a specific and (relatively) simple domain. The domain is not programmable other than through modification of the basic control parameters, and so experimenters must ignore issues that the basic domain does not provide for. Tileworld provides a good domain for testing some aspects of intelligent systems, and especially for testing the basic approach that a more complex agent would exemplify (e.g. testing a reactive algorithm with a specific configuration of the Tileworld before developing and examining a more complex agent based on the same principle in a more complex domain). It is not well-suited, however, to simulating everyday domains at a high level, or to examining more complete, complex agents.

6.3.3. MICE

MICE (Michigan Intelligent Coordination Experiment testbed) is a system specifically designed to simulate the interactions of multiple agents [Durfee et al., 1989], and has many analogies to Tileworld. The MICE simulator is also based on a grid, although the domain is in some ways much less constrained. Agents can move about the grid, create and destroy other agents, and constraints can be defined on the environment and on agents themselves (each agent is user-implemented as a LISP procedure). The result is a very programmable system that can implement many domains within the same grid-based system: Durfee et al. [1989] illustrate the use of the MICE simulator on a predator/prey environment, a fire-fighting environment, and in simulating robot coordination.

Unlike previous simulators, a clear and concise interface is made between agents and the simulator. A complete set of possible instructions that an agent can give the simulator is provided, and the type of information an agent is allowed to receive is made explicit when designing the agent. Also unique is the fact that some effort has been spent in considering the relationship between actions committed to by the agent and their effects manifested over time by the simulator. However, in order to keep the simulator simple, the nature of actions is made somewhat awkward. Each action and event is considered to be discrete, regardless of the amount of time that action or event requires. Thus if an agent moves from one square to the next, and that movement takes four time units, the agent completes the move in the first time unit, and then must wait three more time units until something new can happen to it [Montgomery et al., 1992].

MICE is in some ways a much more general simulator than Tileworld or Phoenix. It is the first that is designed to be explicitly programmable: a user can develop a LISP function to mimic the behaviour of a given agent, using guidelines provided by the system, and can fairly easily link this function into the simulator as an operating agent. It is more complex to use than Tileworld, because it is more modifiable and the domain is less constrained. The interface between the agents and the domain is explicit, however, and many features are also provided to ease the programming of agents.

MICE is an improvement over previous simulators in many ways. However, it too has difficulties. For example, the physics of a MICE domain are entirely concerned with the interactions between agents, and no other environmental processing is performed. This severely restricts the ability of the simulator to support dynamic events, and from this perspective MICE is much more

restrictive than Tileworld [Pollack and Ringuette, 1990]. MICE also shares many of the difficulties of previous simulators. The system is designed specifically for performing coordination experiments, and thus much of the programmability of this system is oriented in this direction. Most of the events that can occur in environments implementable in MICE involve what happens when agents bump into one another, *capture* or *link* to one another (generic terms with specific meanings depending on the domain), and in fact the types of actions the agent is allowed to take are fairly limited and simplistic. These bias MICE to a fairly specific class of applications, as well as a fairly specific set of control criteria.

Perhaps more significantly, MICE shares Tileworld's difficulties in simulating complex domains. While the domains of MICE[92] and Tileworld certainly display some level of complexity, this complexity is confined to specific dimensions of the domain. Spatial complexity in both systems is obvious, for example, but other aspects are far too simplistic: agents do not have the rich abilities for gathering information that they do in the real world. There are no real limits on the amount and type of information the agents can obtain, and even the objects in the world itself (tiles, nondescript symbols representing predators and prey) are organized to keep the "information complexity" of these domains low.

6.3.4. Ars Magna

Ars Magna is a recently developed simulation system designed to simulate the actions of a mobile robot. The system is accurate in that the operations supported by the simulator reflect the abilities of current robotics technology [Engelson and Bertani, 1992]. The system allows the user to define the behaviour of a robot, and provides an extensive collection of functions that allow the robot to interact with the rest of the domain. The domain consists of an indoor area mapped by the user (i.e. a collection of rooms and walls), a collection of user-defined objects, and containers that can hold these objects. This is an extremely sophisticated domain in comparison to those examined previously: the domain is informationally as well as spatially complex, and sensing is limited by the abilities of the cameras mounted on the agent. Unlike Tileworld and Mice, agent-independent events can occur through the

---

92 While MICE is not associated with a single domain, the types of domains to which it is well-suited clearly fall into a small category (as previously described). For the sake of comparison, we will treat this category as the "domain" of MICE.

use of user-defined procedures known as *kickers* that execute randomly. These essentially implement the random agents described in Section 6.2.

Ars Magna will no doubt prove to be the most useful of the simulators examined thus far. Because of the choice of a robotic domain, it directly addresses the issue of avoiding hardware unreliability, one of the primary reasons for making use of a simulation system. By programming the simulator with the abilities of the robot the intelligent system is ultimately intended to control, its abilities can be tested without relying upon (or risking) physical hardware components.

The system can also construct fairly complex variations on the basic robot problem-solving domain. The authors differentiate between two types of variability in the system: *flexibility*, which entails the ability of the system to implement a wide variety of problems, and *tunability*, which entails the ability to modify system parameters within a given problem. Ars Magna provides a great deal of domain tunability, in that the speed of the robot, the abilities of its cameras, etc., may all be set via system parameters. The domain is also flexible, in that the user can define new types of objects, and actions that can be performed using them.

This flexibility and tunability, however, is entirely within the domain. The simulator itself is neither tunable nor flexible. While many parameters are provided, none of them control the simulator itself. The basic timing of the simulator and the rate of change in the world, for example, are not modifiable. Similarly, little in the way of domain flexibility is included. It is unclear how one could implement an entirely different domain in Ars Magna, although the authors claim that the indoor robot simulator can "probably approximate" other robotic environments as well [Engelson and Bertani, 1992].

Ars Magna is intended to be exactly what it is: a useful simulator designed to accurately approximate an indoor robotic environment. It was not designed as a generic simulator and does not function well as one. It fails to meet many of the criteria outlined above; for example, it does not support multiple agents (although its kicker procedures could be used to implement the effects of another agent, there is no way these could be used to simulate explicit agent interaction). The internal mechanisms are largely left undescribed, in favour of extensive descriptions of the interface functions providing the robot with access to the rest of the simulated world. Neither of these points is a direct criticism of Ars Magna as a robotic simulator; however, they both serve as evidence that Ars Magna's main contribution is its accurate domain rather

than its abilities as a generic simulator[93]. Even within Ars Magna's domain, the high level of detail provided may be more than is desired in many applications.

6.3.5. Special-Purpose Simulators

In addition to the simulators described above, many other systems exist within the scope of a single application. Given that none of the simulation systems examined so far are truly generic, differentiating special-purpose systems from those already examined may seem unusual. However, the simulators that fall into this category specifically exist within the context of one system and make no claims to be used in any other manner.

Two of these are especially notable. The *Toast* system [Agre and Horswill, 1992] includes a very elegant and simple simulation system designed to present a kitchen domain to an intelligent agent. The simulator is not easily adaptable to agents other than that for which it was designed, since it relies closely upon a particular and unusual organization for the physics of the domain: every possibly transition for every object is stored with the object itself. However, it illustrates that a simple simulator can handle the information complexity and (to some degree) the timing complexities of an everyday environment. A similar effort has been made by Hammond to provide simple but adequate simulation domains for the *Trucker* (delivery) and *Runner* (errand-running) projects [Hammond, 1989; Hammond et al., 1990]. Agents in these environments interact at a high level (e.g. an operation might be *turn left at 42nd street*), and manipulate objects in the environment in a similar manner. In the case of Trucker, multiple agents are supported in a manner similar to Phoenix, and the interface itself is also quite complex: agents parse information given by the simulator to simulate the sensing of higher level objects (e.g. a building with windows and a lighted sign, sensed as separate entities, might become a store).

The domains presented by these simulators are in general much more rich and complex (although much less well-understood) than those of Tileworld

---

93 In fact, the authors admit that the focus of the project is providing the domain, and most of the existing documentation of Ars Magna [Engelson and Bertani, 1992] is concerned with describing the intricacies of this domain and comparing Ars Magna to other simulators from the point of view of domain. Little is available on the internal operations of the simulator itself.

or MICE. These simulators compensate in part for this complexity by examining the domain at a relatively high-level. Toast, for example, reasons about a great many different classes of objects, using high level operations (e.g. the agent might move a plate from one spot to another in a single operation). This contrasts with the domains of previous simulators, which concentrate on the detailed processing of a few fairly simple objects. This is a difference in granularity, a concept mentioned previously.

The amount of detail a simulation system can provide is extremely variable, ranging from high-level approaches such as those described above to methods detailed enough to precisely mimic complex physical processes. Rather than representing a *move-plate* action in the manner described above, for example, one might minutely simulate each detail of the move, resulting in a much more complex simulation, but also one with much greater accuracy. Theories of qualitative physics (e.g. [Forbus, 1984]) can be used in understanding and simulating complex physical systems in this manner, and such systems are widely available for use in simulation (e.g. [Kuipers, 1986]). However, it is extremely rare that this level of detail is required for examining an intelligent agent and the processes that constitute its behaviour: much of the reasoning an intelligent agent is involved with in commonsense domains is high-level, abstract reasoning.

Because of this, the utility of developing a simulator that specializes in low-level detail for the purposes is examining intelligent agents is low. Even Ars Magna, which provides the most detailed domain of the simulators described in previous Sections, abstracts many details of a real robotic domain in the interest of expediency [Engelson and Bertani, 1992]. The "generic" simulators examined in previous Sections form a middle ground when compared to qualitative physics systems, but the argument still applies. None of the above systems are well-suited to dealing with high-level approaches to information-complex domains in the manner of the special-purpose simulators presented earlier.

The major contribution that these special-purpose simulators offer is their perspective: the view that complex, information-rich domains can be simulated at a high level for the purpose of testing intelligent agents. They are presented here specifically for that viewpoint, and are not critiqued. The criteria used to evaluate simulators in previous systems cannot be applied here, because the criteria themselves depend heavily on the assumption that a simulator should be as generic as possible. Fair comparisons thus cannot be drawn.

6.3.6. Discussion

Each of the above systems has many individual strengths and weaknesses. Considered broadly however, these simulators may be divided into two groups with respect to their abilities to serve as a generic testbed for intelligent agents. Some systems offer complex simulations of specialized domains (e.g. Phoenix, Ars Magna). Such simulators are often designed as part of a specific agent architecture or domain implementation, and serve well in that capacity. However, they generally prove inadequate when an attempt is made to adapt them to a new domain or to a new type of agent. Other simulators (e.g. Tileworld, Mice) are more general or abstract, but provide a great deal of variation on one (usually simple) class of domain. These simulators are adequate for testing simple strategies in problem-solving, but are inadequate for representing more complex domains. Of the four, MICE would likely be considered the most generic, and has been labelled as such by others [Hanks et al., 1993]. However, as Section 6.3.3 has indicated, it has many problems itself.

In fact, while each of these simulation systems is well-suited to some environments, none provide the features necessary to make them truly generic from the point of view of intelligent agent research. Adapting any of these simulators to an agent design or domain that it is not biased toward would likely require a major effort. This is unsuitable from the point of view of developing intelligent agents, as well as from that of studying simulated domains. Agent architectures and domains are highly likely to change during development, sometimes drastically, and if a simulator cannot provide support for such changes, users are likely to spend most of their time adapting one system or another to their changing designs. The only alternatives currently available to this are adapt one of the many special-purpose simulators that have been developed in conjunction with particular agent designs (e.g. [Hammond et al., 1990; Agre and Horswill, 1992]), or to make use of other forms of simulation, such as qualitative physics systems (e.g. [Kuipers, 1986]). Taking the former route involves exactly the same problems as adapting the pseudo-generic systems described above. The latter choice, on the other hand, provides detailed simulation of a particular environment, but none of the specialized features for integrating intelligent agents within a simulated environment.

The difficulty inherent in applying existing simulation systems to the task of examining a broad range of agents in various domains has led to the development of *Gensim*, a generic software testbed for intelligent agents that addresses all of the criteria presented in Section 6.2. The remainder of this Chapter describes the Gensim system in detail. The primary motivation in

the development of this system was to provide simulation for several projects at the University of Manitoba, including this research. After examining the prospects of modifying the systems described above, it was decided that it would be more fruitful to design a new system that could be used for ongoing development of agent architectures and domains in which to examine these architectures. The study of existing systems also pointed out an obvious need for such a generic testbed: a system satisfying all the requirement presented in Section 6.2 would bridge the gap between the two classes of simulators described above, providing a basis for future research in the design and evaluation of intelligent agents.

6.4. Gensim

Gensim is a LISP-based object-oriented simulation system designed explicitly for the testing and examination of intelligent agent designs. Rather than simply providing a flexible domain, as is done in Ars Magna or Tileworld, Gensim provides flexibility in the simulation process itself. That is, rather than providing extensive domain parameters, Gensim provides the parameters and functions necessary for the user to define their own domain and interfaces for agents. This requires greater initial effort on the part of the user, in order to define or modify a domain, but makes Gensim much more widely applicable than any of the simulation systems described above.

Having been designed with generality in mind, Gensim provides the features described in Section 6.2, including:

- The ability to define multiple agents, each of which may consist of multiple processes.

- A clearly defined relationship between agent and simulator, including a model of timing for temporally extended actions.

- A concise and simple object-level sensory and effectory interface between agents and the environment. This interface is provided for convenience, and may be diverged from if desired.

- The ability to control many aspects of the simulation itself, such as aspects of timing, in addition to general domain parameters such as the bandwidth of an agent's senses.

- A modular design, making it simple to take the basic simulator and program any additional features required to support a given agent or domain structure.

A high-level overview of Gensim is illustrated in Figure 6-1. The simulator itself manages an environment, represented as a collection of objects. This simulation environment is objective: it is the universal set from which all intelligent agents' perspectives are defined. The simulator also possesses a collection of procedural knowledge describing the actions that agents can perform on these objects, as well as the physical events that can occur to these objects outside of the influence of any agent.

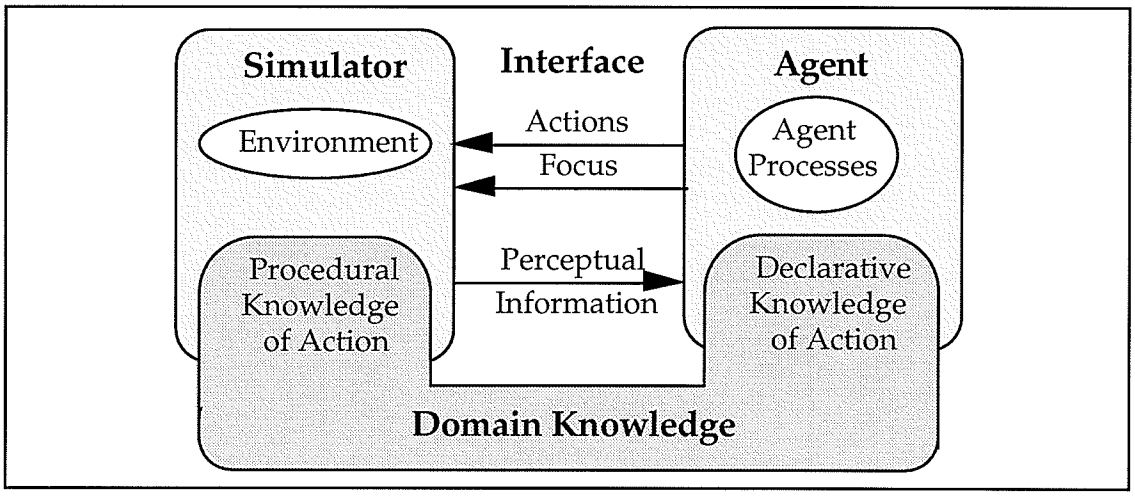

Figure 6-1. The high-level structure of Gensim.

A collection of agents is also defined, each with its own view of the environment based on its sensing ability and memory. The reasoning abilities of an agent are implemented as a set of timeshared processes, that collectively allow the agent to perceive the environment around itself and act on the basis of those perceptions. As each action is carried out by the agent, its effects on the world (both short-term and long-term) are manifested by the simulator. Agents are viewed as "black boxes" by the simulator, in that Gensim neither knows nor cares how the agents arrive at their decisions for action. As indicated in Figure 6-1, some commonality in domain knowledge is required in order for agents to interact with the simulator. For example, the actions that the agent can select from (represented largely in declarative form) must be identifiable by the simulator, which then uses its own (largely procedural) knowledge of action to modify the environment appropriately. Agents also inform the simulator of their interests in objects in the

environment (their *focus*) in order to provide appropriate sensory feedback. However, emphasis is on making agent's knowledge distinct from that of the simulator, and the illustration in Figure 6-1 should not be interpreted as implying that agents must somehow physically share the simulator's knowledge. Rather, the simulator provides an objective description of the domain, consisting of all the objects in the environment (including the agents themselves), and causal knowledge of how those objects interact with one another. Each agent maintains its own (usually limited) perspectives of this environment, much as the real world operates.

The basic design goal of Gensim was to support as wide a variety of agent and domain designs as possible. To this end, Gensim was designed to keep agent and simulator knowledge as separate as possible, and thus limit the knowledge each must have of the other. A complete environment is defined for the simulator, and agents are expected to have their own knowledge of the objects in the domain and how they operate. This is more complex than simply allowing an agent to share the same internal objects manipulated by the simulator, but allows much more flexibility in implementing complex domains. In particular, it allows agents' perspectives to differ, and limits agents' knowledge to that information the modeller wishes them to possess.

In spite of this design goal, it is impossible to ensure that a simulation system can support *any* domain or agent design. Certain basic design concessions must be made in order to use a domain or agent design with Gensim, just as concessions must be made for any two systems to operate together or communicate with one another. In particular, the domain must be organized to obey the timing principles on which Gensim is based.

6.4.1. Timing in Gensim

As mentioned in Section 6.2, the operation of the intelligent agents in a given environment should proceed in parallel with the changing environment. In the real world, questions of timing do not arise: the world is objective and completely independent of the agents that inhabit it. The world runs at a given speed, and the agents follow along at their own pace. While this would be ideal, Gensim is currently implemented on a serial machine, and thus must simulate parallelism using timesharing. Gensim supports multiple agents, each of which may consist of multiple processes, and performs its own timesharing of those processes. The simulation of agent operations is done by continuously cycling through each process of each agent. As agent processes run, they collectively perform actions and make requests for perceptual

information from the simulator. After each process of each agent has been executed, the simulator updates the environment based on the actions of the agents (and other independent events), prepares sensory information for each of the agents based on this modified world, and cycles through the agents again.

In simple simulators (e.g. [Agre and Horswill, 1992]), the basic time interval around which the system is organized is often the time it takes an agent to decide on an action to carry out. Each action in such a system is thus based on information gleaned from the world at a single point in time. In the real world, however, sensory information arrives continuously: by taking its time, the agent can get many views of the environment on which to base its decisions for action. The agent pays for this ability proportionally through the risk of unanticipated and unrecoverable change in the world around itself. Gensim supports this by organizing the system's timing around the interval at which sensory information is presented to the agent. For the purposes of this discussion, this interval will be called P.

By designing the system around the time required to gather information about the world, an agent can be "interrupted" with new information about the world every P seconds[94]. This interval is one of the standard tuneable parameters in a Gensim environment, and is static throughout a simulation. The amount of time a given agent is allowed to deliberate its choice of action (the number of P intervals) is not limited by Gensim in any way. Each P interval gives the agent further sensory information, and the agent can deliberate for as many intervals as desired, facing the consequences of change and missed opportunities in the simulated world. This discrete sensing interval is not unrealistic. Human vision, for example is not as continuous as it might seem on the surface. Experiments have shown that change over intervals of approximately sixty milliseconds or less are perceived as continuous by human vision [Graham, 1965]. So long as a simulator presented sensory information at a rate greater or equal to this, a series of discrete "snapshots" would be indistinguishable from a continuous process by a human and could be accurately treated as such. This same principle can be applied to computational agents.

---

94 I have elected to use seconds as the basic unit of time when discussing Gensim. This in no way implies that Gensim is limited to this time unit as a basis. The design applies equally well to millisecond-level timing as minute-level timing and beyond.

In addition to the sensory interval P, a simulation system for intelligent agents must be concerned with two further timing cycles. The first of these is A, the rate at which an agent commits to actions. In any realistic environment, A cannot be regular: an agent must be allowed to react immediately or take as much time to deliberate as it desires. The other is E, the rate at which changes are made to the environment by the simulator. In the real world, change occurs continuously, but by the same argument used when defining the P interval, a small discrete interval is indistinguishable from a continuous process by a rationally-bounded agent. In an objective simulation of a given environment, this E interval directly affects the accuracy of the simulation. By the principles discussed above, the longer the E interval, the less continuous the simulation will appear to be. However, since in the case of Gensim the simulation exists strictly for the benefit of one or more agents, no change need occur except at those times when information is presented to those agents. Because of this, E is a fixed length interval equal to the shortest agent P interval in a particular simulation.

The relationship between these cycles is illustrated in Figure 6-2. Here, the agent deliberates about and performs four actions. During the course of these deliberations, new sensory information is presented every P seconds. The simulated world is updated every E (= P) seconds, reflecting ongoing changes in the environment.

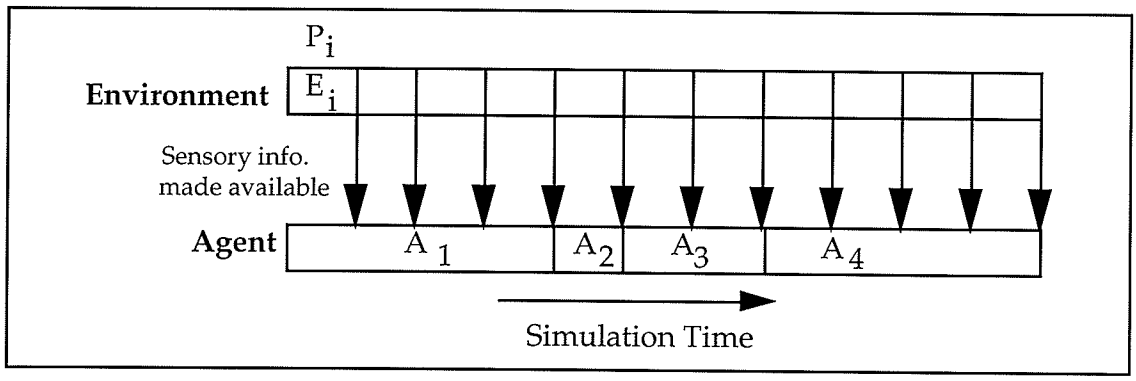

Figure 6-2. Parallel agent-environment timing.

Figure 6-2 represents the ideal timing situation in a parallel simulator where both the simulated world and the agents it supports operate in real time. In Gensim, however, two aspects of this timing differ. As mentioned previously, Gensim is implemented on a serial machine, making it impossible for the agents and environment to run in parallel. Also, while agents must operate in real time, an environment may consist of many

hundreds or thousands of objects, and it may simply be impossible to maintain the state of all those objects in real time. Restrictions could be made to force the simulator to operate in real time, but these would at the same time severely restrict the range of environments that can be implemented.

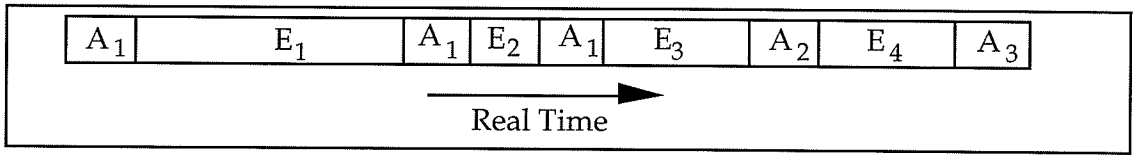

Figure 6-3. Timeline showing interleaving of agent/simulator processes.

These complications are dealt with in Gensim through timesharing, allowing each agent process to run for precisely the length of one P interval. After each process has been run, the environment is updated (based on the actions of agents as well as independent events that have occurred). A new set of sensory information is then provided, and agents may continue their own processing. This timing, for a single agent with a single process, is illustrated in Figure 6-3. Here, the agent processes are run repeatedly, each time for the length of one P interval. An agent may carry out an action during that time, or may deliberate for an arbitrary amount of time. In the example shown in Figure 6-2, the first action the agent takes is deliberated upon for three intervals, the second for one interval, etc. After each E interval, the environment is updated. These E intervals are regular with respect to simulation time (in that each represents the changes that occurred during the previous P interval) but are irregular with respect to real time: the environment may change significantly one cycle, and only trivially the next, and so the amount of real time it will take the simulator to manifest these changes will vary greatly.

The next time the agent process is run, it has new sensory information available to it based on the environmental changes that occurred. The agent may, of course, simply choose to ignore this new information and continue on with what it was doing previously. Agent processes are implemented as LISP functions, and users can explicitly specify that a given process is to run for some fraction or multiple of an P interval if this basic timing is not sufficient. Figure 6-3 shows only one agent interacting with the simulator; however, the same basic timing applies to multiple agents, or agents with multiple processes. All are granted specified time-slices, representing one single unit of time being shared by all agents, following which the simulator updates the environment appropriately.

I believe this timing to be adequate for most domains. It can be criticized, however, on one significant point: since the processes of an agent (and indeed, the agents themselves) are serially timeshared, they cannot simulate some of the interactions possible in truly parallel processes. For example, an agent might consist of two processes: one to recognize objects in which it has an interest, and another to decide what to do based on the objects it has seen. When these run in parallel, the acting process would process input from the recognition process as it became available. This is a much more complex interaction than that possible under timesharing, where the recognition process must recognize all the objects it can in a given interval and then pass them all to the acting process. This is a legitimate criticism, but like the accuracy of the sensory interval, the effects of this depend entirely on the interval size. Like other applications of timesharing (e.g. operating systems), the smaller the time-slices involved, the more transparent the timesharing will appear.

6.4.2. Managing Environmental Change

Now that the basic timing of Gensim has been described, the operations that occur during agent and simulator time cycles can be examined. As stated in previous Sections, the primary purpose of a simulator is managing change in a virtual world. In Gensim, this world is modelled using an independent frame-based knowledge representation system[95]. Each physical object in the domain is described by a frame [Minsky, 1981], and these frames are organized in a hierarchical fashion. The system also supports message-passing, procedural attachment to frames, a form of multiple inheritance, and the use of multiple frame hierarchies. The ability to manage multiple frame hierarchies is critical in a multi-agent simulator, since the simulator must keep track of the objective world and allow agents to maintain their own models of the world if they choose to do so. The use of multiple frame hierarchies allows each agent to make use of the same knowledge representation system used to manage the objective environment. Each agent can also make use of any other knowledge representation system it requires. The various attributes recorded for each object in the simulated world will, of course, differ by domain. However, the system assumes that each object has a given *location* attribute, and that the locations of all objects

---

95 This system was originally developed by Dr. Mark Evans for teaching purposes, and was modified by myself to provide additional features required by Gensim.

are recorded in a similar format. Since the hierarchy is organized primarily by class, retrieval by class is always an efficient process. I recognize that retrieval by location may also be common in some domains, and provide the option of indexing objects by location.

Change in the simulated world is managed through two facilities: agents perform *actions* that can alter the world; and *events* (which may or may not be independent of an agent's actions) may also occur. An event is the occurrence of change in the domain from an unspecified source. For example, a ball may hit a wall and bounce off; a toaster may pop; or the wind may knock something off the shelf. Note than none of these examples is *directly* attributable to any agent: for example, the ball may have been thrown by an agent, but from the point of view of providing an accurate physics for the domain, this no longer matters. An event is defined as a block of environmental changes that occur over a simulator interval. Events may propagate over time into an *event series*, which allows the simulator to represent continuous change over time. The simulator manages an *event queue*, each entry of which consists of a point in time at which an event is to occur, and the name of a routine that will cause the desired changes in the domain to be manifested. Functions are provided for creating random elements in an event, and for inserting new events in the queue from within an event (creating an event series).

Events are defined procedurally and are attached to the domain objects they affect (this implementation will be described in greater detail presently). A ball, for example, may have a *travel-through-air* event defined for it, which moves the ball through the air in a given direction at a particular speed. During the length of time the event runs, it may reduce the speed of the ball and check to see if it hits something in the environment. In either case, the *travel-through-air* event will insert a new event in the queue (to move the ball further along the next time the simulator runs, or to make the ball stop or bounce if it has hit something). Events may interfere with one another: in the above example, a ball might strike another ball and divert it from its path. Once again, this is part of the physics of the domain to be simulated, and routines are provided to dynamically insert, modify, reorder and delete event queue entries to allow the user to implement this physics.

The accuracy of a domain depends a great deal on how events are implemented. Since each update of the world represents a specific time interval, and since that interval can change from run to run, an event should be designed to utilize the time factor involved rather than simply performing the same discrete operation regardless of how much time is used. For

example, an event might move an agent one "unit" in a given direction on a grid, or it might move an agent at a given velocity for the amount of time available. The former results in a poor simulation should the unit of time vary from run to run, while the latter is unaffected. As mentioned previously, the amount of control allotted to the user by a simulator such as Gensim makes it possible for poorly-designed simulations to be designed as easily as well-designed simulations. It is up to the user to use the facilities provided in a manner appropriate to the domain and to the level of accuracy desired.

Actions are closely related to events. Unlike events, change induced by an action has an explicit source: an action is performed by an agent with the intent of accomplishing some objective. Actions in Gensim consist of three components: an intention component, representing the agent's internal justifications for and expectations of the action; an agent-causal component, consisting of the immediate physical actions the agent performs in order to bring about its intentions; and a domain-causal component, consisting of the changes that occur later in time due to time delays or physical interactions with other objects in the environment. Thus a ball being thrown by an agent and breaking a window consists of the agent's intention to throw the ball (the reasoning and justification behind its decision), the actual arm motion that causes the ball to be thrown, and the course of the ball, culminating in its collision with the window. The agent knows only of its intention[96] and the motions that it itself carries out; knowledge of what happens after the ball leaves the agent's hand is entirely dependent on the agent's perception (seeing what happened) and/or the accuracy of the agent's knowledge of the physics of the domain (predicting what will happen).

The simulator is concerned with manifesting the physical effects of an action on the environment. Thus, the simulator itself is concerned only with the immediate changes that would have taken place during the cycle in which the agent performed the action (the agent-causal component of the action), and a series of future events the action sets in motion (the domain causal component of the action). The latter are inserted into the event queue to

---

96 The term *intention* here has nothing to do with the knowledge structures described in Section 5.2.1. Gensim was developed independently of the Waffler architecture, and there is no explicit bias in Gensim to a Waffler agent in particular. Here I mean only that an agent has only its own internal knowledge about what it thinks this action will accomplish and how to go about this action, and that this knowledge is independent of the physical changes that will actually occur in the simulated environment.

have their effects manifested at the appropriate time. An action can cause any number of future events to occur; however, each one of these events must be independent of one another. For example, an action like *throw* performed by an agent might cause the ball to be thrown through the air and through a glass window. The *throw* action should not set up this series of dependent events. Rather, it should simply cause the ball to leave the agent's hand in the current time interval and have some velocity in a given direction. The implementation of the action can then set up a *travel-through-air* event, as described above. This event can in turn propagate itself, and the resulting sequence of events implements the desired behaviour. A sequence of such events is shown in Figure 6-4. An agent throws a ball, which travels-through-air during a number of intervals, hits something and bounces, and will eventually come to a stop. Note that only one of this series of events will be in the event queue at a time, since each one spawns the next. This sequence is also non-deterministic: as the future unfolds, existing scheduled events can be modified or cancelled by other actions and events through the use of the event facilities described earlier.

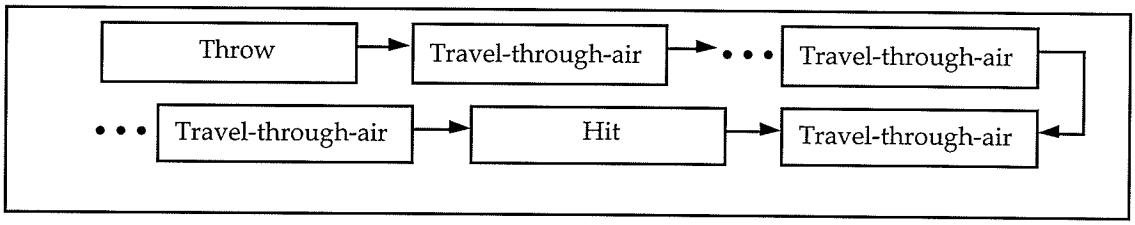

Figure 6-4. An action spawning events in the event queue.

As per events, the representation of actions described here is strictly the perspective of Gensim, and *not* that of any particular agent. Each agent can and generally does have its own representation of the actions it can perform, and will naturally have its own expectations of what carrying out the action will do to the world (these are encompassed by the intentional component of the action described earlier). This knowledge is stored in the agent's own separate knowledge base, along with other knowledge of the domain. The actions defined as part of the environment, on the other hand, define the objective physics of the domain independent of any agent. As each agent process is run, the particular agent performs actions and records its performance in a data structure accessible to the simulator. After all agents have been run in a given cycle, part of the task of the simulator is taking all these actions and updating the environment based on their effects (by invoking the simulator's own procedural representations of these actions through message-passing). Some effects will be as the agents anticipated,

while others (due to random ill effects, misinformation on the agent's part, or interaction with other actions on the same cycle[97]) will not.

The manner in which Gensim views the connection between the actions of an agent and the temporally-extended effects of those actions on the environment is very different from that of any of the simulation systems described above. In the past, simulators have been described as *carrying out* or *executing* the actions of an agent: such phrases invite an inappropriate comparison between the agent-simulator relationship and the relationship between and classical planner and its executor.

The real world has almost no comparison with such an executor, and neither should any simulator intending to model it. The real world is not a servant that carries out the commands of a disembodied agent. Rather, the agent physically participates in an ongoing interaction with the rest of the objects the constitute the world. A simulator, it follows, should provide an environment that can be changed, rather than serving as a method for accomplishing this change.

In keeping with this view, Gensim does not allow agents to *instruct* the simulator in any way; rather, agents commit to and carry out actions during the time periods in which they are active (i.e. during A_i timeslices shown in Figure 6-3). When an agent commits to an action such as *Throw the ball*, it is *not* viewed as a *throw* instruction that is carried out by the simulator, returning some result to the agent. Rather, the agent is viewed to have carried out the agent-causal portion of the action during the previous Ai interval. The simulator, having been given knowledge that the agent has performed this action, in turn alters the affected parts of the environment during the time interval in which it is active. That is, the simulator changes the environment based on the agent-causal portion of the action, and continues the changes indicated by the domain-causal portion of the action over time.

---

97 Currently, actions are not timestamped to indicate the time in the current cycle at which they are carried out. Thus interactions are not always as realistic as they should be - the environment is updated based on all of agent 1's actions during the current cycle, then agent 2's, etc. Agent 2's actions can be invalidated by agent 1's, but only because agent 1 has priority - not because one particular action of agent 1's was carried out prior to one particular action of agent 2's. The realism of this approach is directly affected by the cycle length chosen.

This distributed view of action, placing timing of the agent-causal portion within the agent's time interval and timing of the domain-causal portion in the simulator's time interval, emphasizes the dual nature of action: an agent will always have some direct responsibility, with the physics of the environment providing the rest. It also takes care of many semantic problems that were evident in previous simulators. In a timeshared simulator, when an agent "instructs" the simulator to perform some action, not only is it difficult to place a locus of control for the action, it is also difficult to provide accurate timing for the action. When can an agent assume that the action has occurred? When do results come about? Disregarding questions like these lead to the awkward action/event processing of simulators such as MICE, as described in Section 6.3.3. Since the action itself is entirely contained within the bounds of the simulator in such primitive models, the agent itself really has nothing to do with carrying out the action. It simply decides that it wants something to happen, and the simulator magically makes it so. Gensim, on the other hand, allows agents to be viewed as active participants in ongoing interaction with the world, rather than as passive decision-makers.

Emphasis on event-based processing is a relatively recent phenomena in AI, stemming mainly from recent work in reactive architectures. Because of this, most post-STRIPS theories of action (e.g. [McDermott, 1978]) concentrate on reasoning about specific actions in plans, rather than the connection between an action and future events. However, the view of actions and events used by Gensim does have similarities to some more modern theories. In particular, Lansky [1986] presents a view of change as composed of sequences of events. Agent reasoning about change then becomes reasoning about the history of events that have occurred. However, this model concentrates on agent's internal reasoning about change, rather than actually modelling action in a simulator. Since agent's views and representations of actions are independent of the Gensim model, an agent can use this or any other representation internally to reason about action.

This view of activity is also similar to some approaches in temporal reasoning. McDermott [1982], for example, represents event causation using the predicate *ecause(P,E1,E2,RF,I)*, with the following semantics: event *E2* follows event *E1* after a delay in interval *I* unless *P* (a series of exceptions) becomes true. *RF* is an real number representing where in the time interval of *E1* the delay begins, with 0.0 indicating the beginning of *E1* and 1.0 indicating the end. This performs well theoretically and is useful from the standpoint of analyzing causation (again, a function internal to agents not normally useful in a simulator). However, much of the representation is

impractical. For example, consider the requirement of knowing of the length of *E1* in order to implement *RF*: clearly one cannot know how long *E1* will be until it is completed. McDermott's approach also omits some implementational aspects (e.g. multiple effects, links between actions and events, variable I intervals) provided in Gensim.

6.4.3. Perception

> Mere perceiving, then, is a much more active penetration of the world than at first might be thought
>
> - Erving Goffman, *Frame Analysis*

In addition to performing actions, agents also interact with the simulated world through perception. As stated throughout this Chapter, perception is a difficult component of any simulation. On the one hand, it is a crucial aspect of virtually any domain, while on the other, simplification of perception is one of the primary motivators for using a simulator, and the greater the simplifying assumptions, the more limited the simulator will be.

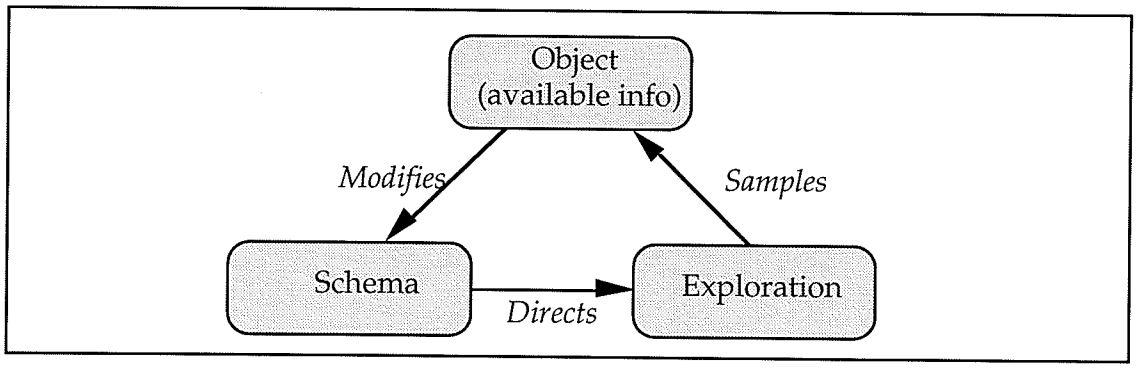

Figure 6-5. The perception cycle [Neisser, 1976].

The general process by which perception should operate in an agent is shown in Figure 6-5. A set of *anticipatory schema* (domain specific knowledge indicating the sensory information in which the agent has an interest) acts as a filter for the vast amount of knowledge available from the world. This knowledge directs the agent to explore the world, looking for given pieces of information. This exploration directs the agent to specific information (objects) in the environment, which then modifies the anticipations the agent has for future sensations [Neisser, 1976]. Within this cycle, one of the basic problems of implementation is that an agent must be allowed to specify its interest in given aspects of the environment, but must also be given access to

information independent of those interests. Perception must exist to confirm the agent's expectations of the world, but also must provide the agent with new information in order to form the expectations of the world that guide its sensory abilities. Neisser [1976] expresses this eloquently:

> We cannot perceive unless we anticipate, but must not see only what we anticipate. If we were restricted to isolated and separate glances at the world, this contradiction would prove fatal. Under such conditions, we could not consistently disentangle what we see from what we expect to see, nor distinguish objects from hallucinations.

Gensim attempts to follow this model as closely as is practical. However, some aspects of perception are simplified, in order to simplify the simulator itself and to conform to one of the basic design goals of Gensim: a simple interface between agents and the simulated environment. One of the major simplifications concerns the internal processing within this cycle. The cycle described above is based on a hierarchical model of vision [Goldstein, 1984], with the environment itself dictating bottom-up processing (processing a visual image into various edges and features), and domain knowledge possessed by the agent directing top-down processing (processing from features to objects using expectations). This is similar to the model of speech understanding implemented in *Hearsay* [Erman et al., 1980]. The major simplification to this model in Gensim is that perception is defined at the object level. That is, an agent senses a combination of complete objects and specific sensory aspects of those objects (e.g. a ball, or the fact that it is red or round), rather than examining the individual edges and features that make up an image of the object itself.

This partially invalidates the hierarchical model, in that the lower-level feature processing that allows the agent to distinguish objects is glossed over. However, removing the complexity of low-level vision from the design of an agent is comparatively the most computationally significant sensory simplification that can be made. In addition, some of the essential character of the hierarchical model of vision is still preserved, in that the information presented to an agent after each P interval is selected based on interests expressed to the simulator at the end of the previous P interval. This preserves the cycle shown in Figure 6-5.

The ability of an agent to express a focus for its sensory abilities is provided in Gensim through a *sensory request* mechanism. This mechanism allows an agent to express interest in certain aspects of its environment, and the methods by which Gensim fulfils sensory requests allows the agent to receive limited sensory information not only about its focus of interest, but also about

objects outside that focus. This provides the balance between anticipation and exploration characteristic of the hierarchical model described above.

Sensory requests operate much as actions, described in Section 6.4.2. An agent makes a sensory request for some particular set of information, and this request is recorded by the simulator. After all agent processes have been run on a particular cycle, the simulator updates the environment according to the actions and events that have occurred during the interval, and then prepares sensory information based on the agents' requests.

Two possible sensory requests are implemented in Gensim[98]. An agent is allowed to explicitly gather information about a given object through an explicitly defined *look* request. An agent can also *scan* in a given direction (recall that directions and locations are part of the definition of a domain) to get a general overview of objects in its vicinity.

Part of the definition of a Gensim domain is the definition of domain-specific aspects of looking and scanning. Like actions, the knowledge required to fulfil sensory requests is defined in procedural form as part of a domain definition and attached to the simulator's knowledge of the agents themselves. The ability of an agent to look and scan can differ from agent to agent. For example, the procedural implementation of *scan* for one agent may result in descriptions of a given number of objects of a certain size (i.e. the agent sees large objects first). A differing implementation may have the agent see all objects within a given distance. The actual methods employed by Gensim to describe objects to agents will be described in the next Section.

Each agent has a finite bandwidth (defined as part of the simulator's knowledge base), implying that they can perceive only a certain number of objects at a time. In addition to this bandwidth, each agent has a limitation on the number of *scan* and *look* operations it can perform in a cycle. The abilities of scanning and looking can also be made mutually exclusive to one another within a cycle. These limitations can be used to enforce bandwidth information very strictly, or to have a fairly loose concept of bandwidth, depending on the domain. If any bandwidth is exceeded (e.g. if there are three looks, and the first two present all available objects, or if there are the number

---

98 Others may be added, but due to the flexibility of these operators, we have not concentrated on further facilities for defining sensory operations. An additional sensory request was defined for the purposes of the examples presented in Chapter 7, and this presented little difficulty.

of looks and scans exceeds the limit), the information is simply not made available to the agent. Note that these bandwidth limitations implement the ability of senses to return information to the agent, *not* the ability of the agent to process that information. The timing of agent processes may enforce a much stricter limitation on the ability to process sensory information than the sensors have to provide it.

In addition to providing visual abilities, we must recognize that perception often involves integrating information from several senses [Neisser, 1976]. This is a point often ignored by simulation systems, but is addressed in Gensim through two additional components to perception. First, an event may cause some noise or other obvious disturbance in the environment. This can be represented through a *signal* event that passes particular facts to the agent (e.g. *signal ring kitchen* could inform the agent that a ringing sound was perceived originating from the kitchen). Signals are passed as is to agent processes when they are run; therefore the agent must have some concept of what the simulator is signalling (in this case, what a *ring* is and where the *kitchen* is). Once again, a limit can be indicated for the number of signals an agent can accept in a cycle, with additional signals being lost. This signalling ability allows limited non-visual perception, and in many domains may be used to reinforce visual perception rather than as an alternative to it.

Finally, limited sensory information at a level higher than the basic object level is available. When an agent carries out actions, it is possible when updating the environment based on those actions to signal the fact that an error has occurred during the course of the action. That is, if an agent picks up a ball, and the simulator decides that the course of change has caused the agent's hands to slip and the ball to drop, the agent may (at the discretion of the developer of the domain) be informed of the fact that an error has occurred, rather than forcing it to look and see that the ball has been dropped. This is somewhat artificial, but is useful in that much of the information necessary to come to such a realization is non-visual (in this case, the agent would feel the ball slip from its grasp long before it could judge by vision that the ball had not hit its target). It is also useful in situations where realizing errors strictly due to vision would be difficult: for example, when an agent bumps into a wall, it is much easier to tell it that it didn't advance, rather than leave it where it is and force it to reorient itself in the world. Again, this is an option that is available when designing a domain. It is not a simplification that *must* be made.

6.4.4. Agent-Simulator Communication

One important aspect of perception and action has been omitted in the previous Sections. For an agent to perform an action using some object, or be able to look at a given object, communication must take place: the agent must somehow describe the object(s) it is interested in to the simulator. If an agent picks up a ball, for example, it must somehow inform the simulator which ball the action has been performed with. Similarly, if the agent wants more information about "that thing over there", it must describe the object as best it can, in order for the simulator to differentiate it from other objects and thus respond with the correct information.

Communication is one aspect of the relationship between an agent and the real world that can never be completely approximated by a simulator. The reason for this is that no communication whatsoever takes place when an agent interacts with the real world (at least in the sense that communication is normally thought of). When an agent wants to pick up an object, for example, it just does so: it doesn't have to communicate this fact to the world. This is in part because the world does not exist for the benefit of an agent, the way a simulator does: any agent is simply an object in the world, like any other physical object. It is also due to the fact that the real world keeps track of itself: an agent's actions physically alter objects, rather than indirectly manipulating some virtual representation of those objects. A simulator, on the other hand, keeps a detailed representation of every object in the world, and changes in the world are manifested by alterations in these representations. Because it no longer occurs naturally, change initiated by agents must be communicated to the simulator in some way. A stronger tie between the agent and the rest of the world is thus necessary.

However, in most simulators, this tie is far too strong. Many simulators do not even use separate representations of objects for agent and simulator: the identical physical chunk of knowledge that describes an object to the simulator also describes it to any agent. When compared to the real world, this is like taking each object in a physical environment, labelling it with a unique agreed-upon symbol, and referencing everything in that fashion. This "trafficking in constant symbols" [Agre and Chapman, 1989] is widespread, unrealistic, and completely unacceptable. Supporting multiple agents already eliminates the use of symbolic labels on all domain objects to provide perception to agents, since each agent must have its own perspective in any useful multi-agent domain. Even if this point is ignored, perception would still be extremely limited if agents were forced to use the same internal object structure that the simulator manipulates. There would, for example, be no

mechanism for limiting access to certain visual aspects, since the agent would automatically acquire all the knowledge of an object maintained by the simulator.

As stated above, there is no complete answer to this problem, as the use of a simulator itself introduces the need for artificial communication. Indeed, the problem of linking perception to internal representation is a severe one, that goes far beyond simulation applications [Etzioni, 1993]. The question thus becomes one of providing these communication abilities in a way that is as unobtrusive as possible to the rest of the architecture of an agent. There are two comparisons that can be made to the way in which humans specify objects in ongoing activity that aid us in this task. Suppose Henry is looking across the surface of his desk, looking for a pen (assuming that there are several on the desk). One could argue that when Henry sees a specific pen he wants, he guides his hand to pick up the pen by referring to it in an indexical-functional or *deictic* [Agre, 1988] manner. That is, the pen he wants to pick up is designated by reference to Henry himself or the objects Henry knows about. Thus the pen may become *the-pen-by-my-left-hand*, or *the-red-pen-by-the-coffee-cup*.

On the other hand, it could be argued that Henry, having looked at the desk, would designate the pen he desires using some internal symbol (the label or designation for this symbol being unimportant, *q1v479* having as much meaning internally as *pen-11*). In this case, Henry could guide his hand by referring to the pen as *pen-11* (or *q1v479*, or whatever): the internal information kept about this pen (estimates of distance from Henry's hand, for example) obtained through vision would guide Henry to the pen.

Both of these methods allow agents to designate objects in a manner distinct from their designation within the simulator. Equally importantly, neither endows the agent with any special ability to access the simulator's objective knowledge. The former method, by requiring the agent to describe an object from its own perspective, puts the onus on the agent to generate a description if it does not already represent objects in such a fashion. This may be inappropriate in time-constrained environments, since an agent may have to spend most of its time generating descriptions. Thus, if multiple agents were to be compared, those that already make use of a deictic representation would be artificially more successful, since they could spend their entire time allotment doing useful work as opposed to generating descriptions of objects for purposes of communication.

Because of this possible bias, Gensim supports both methods of description. The former class is known in Gensim as an *object descriptor*, and is a data structure that describes an object by its relationship to the agent itself or objects the agent knows about. Two examples of object descriptors appear in Figure 6-6. Each object descriptor consists of a header identifying the structure, and a conjunction of clauses. Each clause states some condition that a domain object must satisfy to match the description. These may be standard comparisons (e.g. in the first example, it is stated among other things that the colour of the desired object is red), or may be arbitrarily complex user-defined conditions. When making use of such comparisons, descriptions may be recursive. The second example illustrates this: we compare the colour of the object to that of another object referred to from the agent's perspective (in this case, the ball that shares the same location as the agent itself). This recursion may be arbitrarily deep.

```
(DESC (equal class ball)                    "The red ball that weighs  < 5
      (equal colour red)                     pounds sitting by the sink"
      (less-than weight 5)
      (equal location by-sink))

(DESC (equal location (1 1))                "The object at location 1,1
      (equal colour                          that is the same colour as the
        (DESC (equal class ball)             ball that is where I am"
              (equal location self)))))
```

Figure 6-6. Two examples of object descriptors.

The choice of attributes used to form the clauses of an object description is almost entirely up to the agent. Only one restriction is made, in the interests of computational efficiency. When processing an object description, the initial list of potential objects to which the description could refer consists of every object in the environment. In order to make this initial list considerably smaller, Gensim makes the assumption that one of the clauses will be an attribute under which domain objects are indexed. This is by default a class, and thus a clause describing the class of the desired object is required. As previously mentioned, locations may also be used as an index on objects (the lack of a class clause in the second example in Figure 6-6 indicates this, and also illustrates the type of symbolic location label that is typical of such an index).

The use of these descriptors enforces a strict barrier between the knowledge of the agent and that of the simulator. However, as mentioned previously, the

simulator and agent must have certain concepts in common in order to be able to communicate with one another. In the above examples, both must agree on the concept of *red*, what a *ball* is, and what a *weight* attribute describes. This implementation of object descriptors also requires that the simulator and agents share class names for objects, as well as the same representation of locations. This violates to some degree the desired distinction between agent and simulator knowledge bases. It also involves a relatively minor sacrifice in agent autonomy [Barker et al., 1992]. However, they are absolutely required for any communication to take place: there can be no communication without some agreement on the definition of objects. Overall, I view the physical separation of agent and simulator knowledge as essential, but shared aspects of that representation as equally essential.

As mentioned previously, there are some cases where this method of referring to objects is impractical. In addition to the previously cited example, there may simply be situations where it is desirable to measure the performance of agents without including the overhead of generating descriptions of objects. This is especially useful to those who view this method of communication as artificial, for which there is indeed some argument. In these cases, Gensim provides *object references*, which allow an agent to reference its own internal symbol for an object when informing the simulator of actions or sensory requests. An object reference is a much simpler structure than an object descriptor, consisting only of the agent's symbol for the object and a marker labelling the structure as an object reference. When an agent refers to an object via an object reference, a description is created by the simulator at translation time (that is, when it comes time to update the world based on the action that contains the reference, or to fulfil a sensory request on the same basis). This is done by allowing the simulator access to the agent's knowledge base. An object descriptor is constructed by the simulator based on the agent's knowledge of the object, and is evaluated as described above. Because it requires the simulator to access to the agent's own knowledge base, object descriptions can be used only in agents that make use of the simulator's own internal frame representation system for knowledge representation. Once again, this form of communication violates the basic tenet of complete separation of agent and simulator knowledge, and also limits autonomy, but is required for communication. Note however, that this does not create the same "trafficking" problem described by Agre and Chapman [1990]: here, the simulator is examining the agent's knowledge to see what symbols it uses internally to describe a given object, *not* the reverse. Both methods of communication are necessarily imperfect, but only by virtue of the fact that in

the real world no communication of this sort is necessary. In that such communication is unavoidable in a simulator, I view both of these alternatives as valid approaches.

When the simulator encounters one of these descriptors or references, it attempts to match them to a domain object. There are three possible results of this matching process: the descriptor or reference may match no objects, a single object, or many objects. The first category is considered an error, and the agent may or may not be informed of this as the user chooses. Gensim assumes that an object description or reference should always reference a single object, so the second category is the norm. This is not as restrictive as it sounds; objects being manipulated in a group can almost always be referred to as a unit (e.g. a bunch of grapes, or a container of marbles). To be realistic, operations should also always refer to items in units: I may pick up a *handful* of marbles in a single operation, for example, but I do not refer to picking up 25 individual marbles at once. This leaves the third category open to interpretation. In some situations, an object description or reference that matches more than one object may mean that either is suitable (e.g. if an agent performed a *pick-up pen* action, and referred to the pen ambiguously, it may not matter which the simulator chooses to update. On the other hand, it may be absolutely crucial. Gensim supports either case, through a global parameter that allows actions to choose objects randomly or signal an error upon ambiguous references to objects.

Thus far, I have only discussed the abilities provided by Gensim to aid in communication from agent to simulator. Another vital ability, however, is the ability for the simulator to effectively communicate sensory information to agents. Two of these methods, signals and errors, have already been discussed. However, in many domains, the majority of information communicated to agents by the simulator will be in the form of direct responses to sensory requests. As mentioned above, a primary motivation is the separation of agent and simulator knowledge bases. While the simulator can access agent frames in order to generate object descriptions, allowing direct agent access to simulator frames is unacceptable. Thus descriptions of the objects maintained by the simulator are provided to agents as sensory information. This is performed in a computationally simple manner: in fulfilling the sensory request, the simulator has access to all information about a particular object. The simulator simply singles out all attributes of the object marked as *visible* (part of the definition of an object), and adds them as a group to the buffer containing the sensory information for the agent. Thus an agent might request information about *the-object-in-my-hand*, and be told that it is a white china cup filled with tea. As mentioned previously,

information obtained from all sensory requests are kept in a single buffer, and it is naturally the agent's responsibility to link these descriptions to objects in its own knowledge base or insert them as new objects. Thus if an agent knows about a yellow ball at a given location, and sees such an object at a different location, it is up to the agent to decide if the object has moved or if the latter object is physically distinct from the first.

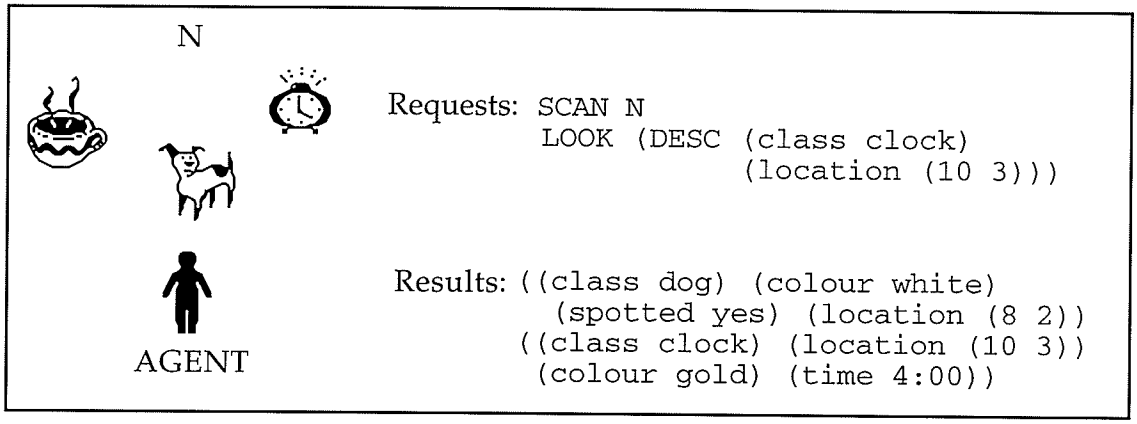

Figure 6-7. Example sensory requests and results.

A simple example of the fulfilment of an agent's sensory requests is shown in Figure 6-7. The agent inhabits a simple grid-based domain, and has three objects sitting in front of it: a cup, a dog, and an alarm clock. In one particular cycle, the agent wishes to scan north, and also look specifically at the clock. The clock is described using an object descriptor. The results of the sensory requests are returned as a list of the *visible* attributes of the objects that fit the sensory requests. This objects are determined by the particular implementation of scan and look for this domain. Presumably, the information about the clock will have been returned by the *look*, and the dog by the *scan*. The coffee cup is not perceived, possibly because the implementation of *scan* omits a certain type of object, or because of bandwidth limitations. In addition, other attributes of the clock and dog which are visible may still be omitted by the scan/look.

6.4.5. Defining a Gensim Domain

Having described the operation of the facilities provided by Gensim, we can now begin to examine how the various entities that make up a Gensim environment are defined. In designing Gensim, I have emphasized the development of common facilities for constructing simulated domains,

rather than specialized facilities for supporting a single specialized domain, as has been done previously. Gensim provides definition facilities (functions and macros) to allow users to easily describe aspects of the domain where names and function will vary (e.g. agents, actions, domain objects). Other aspects of a domain that have a stable purpose (e.g. locations, domain-specific aspects of vision) are provided through user-defined functions with specific names.

There are essentially two aspects to defining any domain in Gensim. First, objects that exist in the domain must be defined, and the simulator's universal knowledge of how those objects may be manipulated (the objective physics of the domain) must be described. After an objective domain is available, each primary agent and its associated perspective of the simulator's objective knowledge must also be defined. Throughout this Chapter, I have emphasized the unique nature of primary agents within a simulation, having a dual role as objects in the environment and primary instigators of change. This dual role is also illustrated in the typical use of a simulator such as Gensim: to test different agent designs in an environment, or to test the adaptability or applicability of a particular agent design to a variety of environments. Because of this emphasis, I have attempted to make the definition of primary agents as independent as possible from that of the environment.

Gensim supports all three of the types of agents (primary, controlled, and random) described in Section 6.2, and all are defined in much the same fashion. Primary agents are the most complex of the three, since both the simulator-related aspects of the agent (that is, the agent as an object in the environment), and the processes and knowledge necessary for the agent to function must be defined.

The former category is accomplished through an explicit *defagent* macro. This facility defines the names of the processes an agent consists of and allows overriding of the simulation parameters that control how long each process runs. It creates a frame partition to hold the agent's knowledge (should the agent wish to make use of the built-in knowledge representation mechanisms), and defines a frame in the simulator's knowledge base to hold information about the physical nature of the agent. It also allows the user to specify state variables, whose values are to be saved and restored each time an agent process is run. Together, the agent partition and this list of variables provide the execution context of the agent's processes. This facility also allows the user to use specific functions to initialize the agent's knowledge base and reset it after each run. Figure 6-8 illustrates an example of this

process: an agent named *bill* is defined, consisting of three processes (*p1,p2,p3*), and a frame partition known as *bill-frames*. Functions for initializing and resetting the agent's knowledge base are names, and the values of several variables (*x,y,z*) will be saved and restored as part of the execution context.

```
(DEFAGENT BILL :PROCESSES '(P1 P2 P3) :PARTITION 'BILL-FRAMES
               :STATE-VARIABLES '(X Y Z))

(DEFUN P1 ()
;recognize and handle errors from the previous cycle
....)
(DEFUN P2 ()
;recognize objects given sensory information from the simulator
...)
(DEFUN P3 ()
;choose an action based on the current situation
...)

(DEFACTION 'THROW 'THROW-OBJECT :ATTACH-TO BILL)
(DEFUN THROW-OBJECT (some-object)
 ;send a throw message to some-object, and update the changes this
 ;action makes to bill
 (SEND 'THROW TO-BE-THROWN)
...)

(DEFEVENT 'TRAVEL-THROUGH-AIR 'TRAVEL :ATTACH-TO BALL)
(DEFUN TRAVEL (DIRECTION)
  ;Procedural aspects as defined in Section 2.2:  make the ball move
  ;in the given direction, and check for collisions, inserting new
events
  ;in the event queue as appropriate
...)
```

Figure 6-8. Agent/action/event definition.

The second aspect of agent definition, the internal knowledge and processing of an agent, is largely left up to the design of the user. Functions must be defined to match those described in the agent definition above, but their internal workings are left unconstrained. For example, the definition in Figure 6-8 is for a universal planning agent used in a sample Gensim domain developed to test the simulator itself. The first two processes recognize errors and objects, essentially integrating sensory information, and the third uses this new knowledge to select an appropriate action from a finite set of possibilities.

In keeping with the timeshared nature of Gensim, primary agent processes are best organized in a specific way. The strictness of this organization depends largely on the type of domain used, but for a real world application, several important principles apply. Since each process is run for a specific length of time, processes should make use of the agent's ability to store intermediate results in state variables, in order to have coherent processing across time cycles. The processes themselves are also best organized as *anytime algorithms* [Dean and Boddy, 1988]: algorithms that produce an answer and improve on it as more time is given. This organization ensures that an agent will have some result from a process even when run for short time intervals. If a process is organized to examine all alternatives before preparing a result, on the other hand, a short time interval may not allow the process to ever get to an answer. None of these are hard-and-fast rules: situations like the latter example may be entirely appropriate for some domains.

As mentioned previously, an agent can make use of any internal representation for action it desires. However, the agents' internal representations of its own abilities (or those of others) is essentially a model of the simulator's own knowledge of action. A procedural definition of each action (possibly specific to particular agents) must therefore exist within the simulator, in order for it to be able to update the world accordingly as agents go about their own courses of activity. These definitions are provided using an explicit *defaction* facility, an example of the use of which is also illustrated in Figure 6-8. The defaction macro names an action (in this case, the action is called *throw*), associates a procedural definition with the action (the procedure *throw-object* in this case), and links the action to a particular agent or group of agents known to the simulator. In the example, this action definition is particular to Bill, but it is also possible to construct an action that is applicable to several or all agents. An abstract view of this structure is shown in Figure 6-9. Bill has two actions which it can perform, throwing an object and moving itself. These are done through message passing, in the same manner as the event processing described in Section 6.4.2. The *throw* action in this case, sends a throw message to an object, and the object can then proceed as dictated by the physics. If a ball were thrown by Bill, the result would be the sequence of events previously shown in Figure 6-4: a string of *travel-through-air* events, ending with hitting another object or running out of energy and halting. Once again, this is the *simulator's* view of the agent's performance of an action: it defines only what the simulator needs to manifest change. Internally, the agent may reason about action in whatever way it desires, from a simple STRIPS formalism, to making use of the

facilities for action provided by Gensim (e.g. an agent could manage its own internal event queue), to custom-designed representations for actions. Gensim makes no restrictions on how an agent reasons about actions internally; only how the agent specifies the actions it has performed in the current cycle to the simulator.

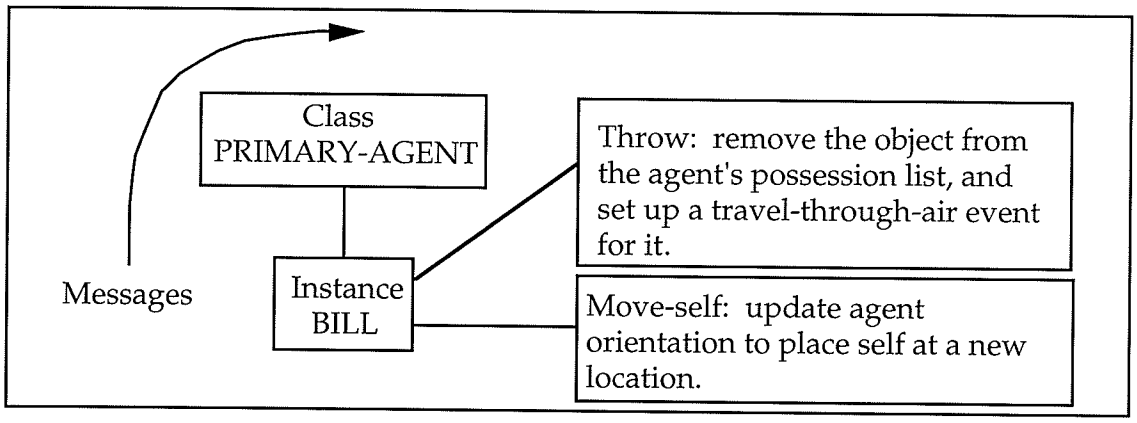

Figure 6-9. Actions attached to a particular agent.

Thus, agent actions operate as follows: as agent processes are executed by the simulator, actions are performed. Once all processes for every agent have been carried out, a cycle is completed, and the environment is updated. One of the tasks to be performed during this update is to reflect change made by the agents' actions. This is done by taking all the actions performed by each agent in turn, and for each action, sending a message to the frames containing information about that particular agent. This invokes the simulator's procedural representation of that action, bringing about change to the environment. The name of the message to be handled is the same as the name of the action (e.g. a *throw* message is sent to the frame defining the simulator's knowledge of Bill after Bill indicates the performance of a *throw* action), and is defined by *defaction*, as is the name of the procedure to manifest the changes that action makes on the environment. The call to *defaction* shown in Figure 6-8, for example defines the name of the procedure used to handle the effects of Bill's *throw* action to be *throw-object*.

As mentioned in Section 6.4.2, events are handled in a similar manner to actions. The only real difference is that the procedures implementing events are bound to objects affected by the event, rather than to an agent. For defining events, Gensim provides an explicit *defevent* facility. *Defevent* takes an event name and binds a procedure to a particular object participating in the event. Figure 6-8 shows the use of *defevent* to define the *travel-through-*

air event used as an example in several Sections of this Chapter, and attach it to an object class named *ball* (presumably defined elsewhere). The structure of event bindings is illustrated in Figure 6-10. Among other things, this Figure shows an event called *travel-through-air* defined for all objects of class *ball* . Like actions, message-passing is used to invoke events. Whenever it is desired to have a ball *travel-through-air*, a message is simply sent to the object involved (when the event was at the top of the event queue). Messages are given to instances (e.g. *ball-1*), and move up the hierarchy until a handler is found. Such a message might be sent when processing the event queue as part of updating the environment, or as part of an agent's actions. The object-oriented approach used by Gensim to implement actions and events is modular and allows the physics of a domain to be defined and extended easily.

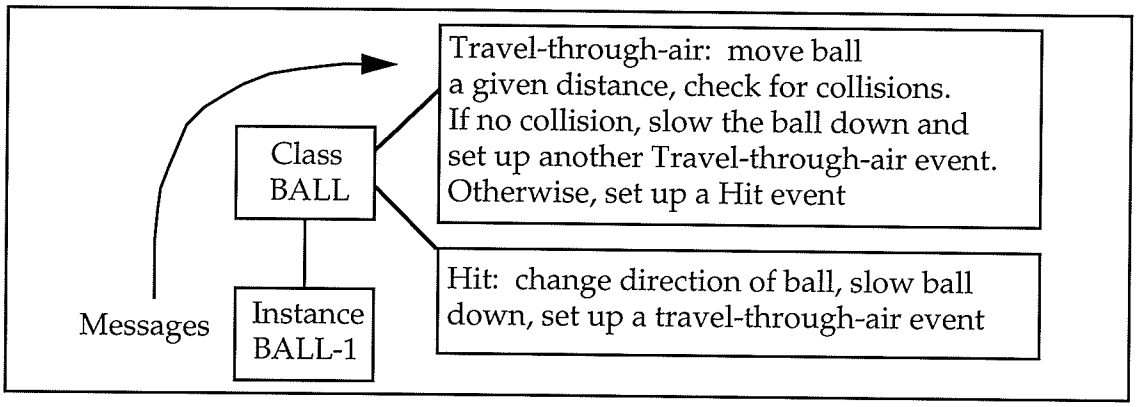

Figure 6-10. Examples of events attached to objects.

The definition of controlled and random agents is provided by similar facilities as those provided for primary agents, but the agents themselves are by their very nature much simpler. Controlled agents require no active processes, as primary agents do. This is because they merely respond to outside events in a completely predetermined manner: they have no deliberative aspects. Recall that simple machines are representative of controlled agents: a toaster, for example, is activated by the user, heats up bread, and sets up a future *pop* event. An example definition of such a controlled agent, using Gensim's *defcontrolled* facility, is shown in Figure 6-11. This facility simply creates a frame within the simulator to describe a controlled agent, and names the agent. The changes that a controlled agent can make to the environment are grouped together as actions, and are

defined using the same *defaction* facility used for primary agents[99]. In the example shown in Figure 6-11, two actions are defined and bound to the toaster: a toast action, which will presumably be implemented to cause any bread in the toaster to get dark and the toaster to eventually pop, and an unplug action, that would presumably stop this process.

```
(DEFCONTROLLED 'TOASTER)
(DEFACTION 'TOAST 'TOAST-FUNCTION :ATTACH-TO 'TOASTER)
(DEFUN TOAST-FUNCTION ()
 ;define the sequence of events necessary for a toaster to operate.
 ;e.g. check that the toaster is plugged in, and if so, set up a
 ;series of events that will culminate in the toaster popping
 ....)
(DEFACTION 'UNPLUG 'UNPLUG-FUNCTION :ATTACH-TO 'TOASTER)
(DEFUN UNPLUG-FUNCTION ()
 ;define what happens when the toaster is unplugged (in this case,
 ;remove future pop event and stop bread from getting browner)
 ...)

(DEFRANDOM 'WIND :ACTIVATE 'BLOW :PARAMETERS NIL)
(DEFUN BLOW ()
 ;go through all objects in the domain, and use a 50% chance on each
 ;of knocking them over.  if an object is knocked over, alter that
 ;fact in the simulator's knowledge base.
 ...)
```

Figure 6-11. Definitions of controlled and random agents.

Like those of other types of agents, the actions of controlled agents are manifested by explicit message passing and procedural attachment to objects, just as they are in primary agents. An agent could send a toaster a *toast* message, for example (simulating activating the toaster), or the same message could be sent internally be the simulator (e.g. if the toaster were active, unplugged, and then plugged back in). There is a semantic difference, however, in that primary agents are viewed as having performed their own actions in pseudoparallel with the simulator, while controlled agents exist entirely within the bounds of the simulator itself.

Random agents, defining random changes to the environment, are defined in a similar fashion, except that the functions associated with such agents are

---

99 The exception to this is that controlled agents are entirely deterministic, have no deliberative aspects, and exist entirely within the simulator.

executed every simulator cycle. Like controlled agents, random agents can bring about immediate change, or can make use of event handling facilities to induce change in the future or alter the course of scheduled events. This implementation differs from the that used by Ars Magna, the only simulator I have examined that supports such entities. Random agents in Ars Magna have as part of their structure the probability that they will run on a given cycle. I assume such randomization to be part of the actions that a random agent may take (that is, built into the function associated with the random agent). Thus the function associated with a random agent like wind, defined in Figure 6-11, would run every cycle, and would have individual probabilities of altering the position or orientation of each object (or classes of objects) in the domain. I believe that keeping the random portions of a these agent entirely within the bounds of their implementation is a much more flexible approach than that used by Ars Magna, since it allows the use of these individual probabilities. However, the affect of large numbers of random agents on the speed of the system is much greater in this approach.

As mentioned at the beginning of this Section, however, some of the definition of the domain falls outside of the automated facilities described thus far. The most obvious example of this is the definition of a physical map of the domain. This is done by first setting a parameter in the simulator to record the valid directions supported by the domain (e.g. compass points, left, right, etc.), and their inverses (if supported). Following that, the map itself must be implemented in a function called *next-location*, which accepts a current location and a direction and returns the next location. This forms the basis for all movement in the domain, and also defines the format of locations. To date, I have used both symbolic and grid-based locations in Gensim domains.

Other aspects of domain definition are defined in much the same way. Standard functions exist for *scan* and *look* (as described in Section 6.4.3), implementing the domain-specific aspects of these sensory requests. Users are expected to modify these functions accordingly to suit their domain. This sounds much more intimidating than the process actually is in practice: the functions that must be defined as part of a Gensim domain rely entirely on other aspects of the domain that the user has already described (e.g. what locations and directions are available), and require little knowledge of the internal structure of Gensim itself.

In addition to the facilities for programming complete domains, Gensim also provides a number of system and agent parameters that can be varied from run to run. These include the length of a process time-slice and various

sensory options for individual agents, described in Section 6.4.3. A number of output and general domain variations also exist, for example: verbosity of output; ability to pause between output cycles for interactive examination of agent's internals; definition of specialized output functions; an option to allow the domain to be indexed by location rather than object class; and options for timing correction when processes are interrupted due to garbage collection in LISP.

6.4.6. Overall Gensim Algorithm

As a whole, Gensim is a complex software system. However, as is evident from previous Sections, this complexity is mainly due to the interaction of its many components. Every aspect of Gensim, from agents' senses to the definition of actions, has been made explicit, and the system is relatively easy to make use of.

```
repeat until system halted
 run each process of each agent
 get all events for this cycle from event queue
 update environment based on events
 update environment and event queue based on controlled agent actions
 update environment  and event queue based on random agent actions
 for each primary agent
  get all actions (translate object descriptions and references)
  update environment based on actions
  return any errors to agent
 end for
 for each primary agent
  get all sensory requests (translate descriptions and references)
  create descriptions based on sensory requests
  return information to agent
 end for
end repeat
```

Figure 6-12. Overall Gensim algorithm.

Now that all individual components have been examined, the overall algorithm of Gensim is easily described. This algorithm appears in Figure 6-12. After initializing the system, Gensim enters a repetitive cycle of timesharing agents. Each agent process is run, during which agents perform actions and request sensory information. Before the world is updated by the simulator to reflect the changes made by these actions, any independent events are taken care of, and controlled and random agents are allowed to manifest their changes. This represents the changes that occur during the

time period in which the agent was deliberating and performing its own actions. After this has been done, each agent in turn is examined, its actions extracted and the environment updated accordingly. Any errors that occur during this process may or may not be returned to the user, depending on the domain. Finally, the sensory requests of each agent are examined, and information based on these requests is prepared and given to agents in preparation for the next cycle.

6.5. Summary

This Chapter has described Gensim, a generic simulation system designed to support the simulation needs of the Waffler architecture as well as a wide range of other projects. Gensim meets all the criteria described in Section 6.2: it is a modular system, allowing the user to easily substitute agents or modify aspects of a domain, supports multiple agents and multiple processes, and the object-level interface between the agent and simulator is straightforward. The user can control many aspects of the simulation through system parameters, and can design a domain to keep track of any aspect not already built in to the simulator. Perhaps most importantly, however, the relationship between an agent and the simulated world provided by Gensim is clear and explicit, unlike previous simulators: details of agent timing, actions and causality, and perception have all been explicitly described. This allows potential users to gauge the fit of their domain to Gensim quickly, rather than attempting to implement the domain and only later learning of semantic details that complicate the implementation. The wide variety of features, coupled with an extendible design, make Gensim applicable to a much wider collection of domains than any of the simulators described here.

Since the focus of Gensim is on providing a set of facilities for constructing and simulating a wide variety of domains rather than on providing a single parameterized domain, the system is necessarily more flexible and complex than previous simulators. Flexibility is required in this design goal, and complexity is a by-product. Gensim gives users the power to flexibly implement the domain of their choice, and is the only one of the simulation systems that can support the use of a single agent architecture in widely varying domains. Even considering a single domain, unless that domain closely matches one of the simulators described in Section 1, a generic simulator such as Gensim will be the only alternative to constructing a complete customized simulator. The price of this flexibility is requiring the user to define all agents, operations, and interactions supported by the domain.

Gensim is implemented in Macintosh Common LISP, and occupies approximately 150k of source code, including its knowledge representation system. The contributions and limitations of this system will be described in greater detail in Chapter 9. The following Chapter illustrates the use of Gensim in creating a domain and a partial implementation of a Waffler agent.

CHAPTER 7

IMPLEMENTING AN IMPROVISING AGENT

Ah, to build, to build!
That is the noblest of the arts.

– Henry Wadsworth Longfellow, *Michael Angelo*

7.0. Introduction

The validity of any scientific model hinges on its ability to accurately represent the behaviour for which it was designed to explain. In proposing a new model, there are two requirements that arise from this: to illustrate that the model does indeed explain the behaviour on which it is based, and that it does so more accurately or extensively than existing models. In the case of the Waffler architecture, these requirements have to a large extent already been met. Chapter 3 has outlined the limitations of the classical and universal planning architectures in general and with respect to the phenomena displayed during the performance of everyday activities. Chapters 4 and 5 have presented an approach and an agent architecture based specifically on the phenomena of everyday activity, and have contrasted the fit of these phenomena to the abilities of previous architectures. In particular, I have shown that conceptually, the Waffler architecture both subsumes the capabilities of a universal planner and explains far more complex phenomena: how an agent can go beyond specific routines and make

use of deeper world knowledge when necessary, allowing the agent to perform everyday activities in a satisficing manner.

The fact that the Waffler architecture and the improvisational approach it embodies can be used to explain these phenomena validates the approach in part. In computer science, however, we have the ability to do much more than illustrate how well new theories work on paper: we can make the theory active through a computational implementation, physically demonstrating both its advantages and its limitations. The power that such implementations bring to the research process in AI is described by Dehn and Shank [1982, p. 356]:

> The building of programs is of methodological importance as a scientific tool for the refinement and debugging of theories of intelligence. Programs reveal hidden assumptions and lack of clarity in the underlying theory. Programs, once written can also serve as experimental apparatus, in order to try the theory out, exploring its explanatory value. Programs also serve as a demonstration of the theories they embody. Programs allow theories to evolve.

In addition to these obvious advantages, there is another reason for presenting this work in terms of a computer program. What has been presented thus far is an *approach*: a conceptual framework from which to view a problem and an abstract solution to it. This framework is by no means detailed enough to construct a complete system for improvising in a complex domain. What is missing thus far are the engineering details that bridge the gap between the conceptual framework and a full embodiment of the architecture in a realistic domain. The issues involved at the conceptual and implementational levels differ greatly. These differences are summarized by Nii [1986]:

> Problem-solving models are conceptual frameworks for formulating solutions to problems. The models do not address the details of designing and building operational systems. How a piece of knowledge is represented, as rules, objects, or procedures, is an engineering decision. It involves such pragmatic considerations as "naturalness", availability of knowledge representation languages, and the skill of the implementers, to name a few.

In order to further demonstrate and validate both the improvisational approach and the Waffler architecture, and to serve as a basis for future research, this Chapter describes a modest implementation of a Waffler agent engaged in episodes of an everyday activity in a simulated environment. Many of the techniques employed in this implementation were developed during the course of a more primitive prototype implementation described in [Anderson and Evans, 1994]. This prototype consists of a number of agents of various types (including simple improvising agents) inhabiting a predator-

prey environment and balancing the need to obtain food with the need to seek shelter from pursuing predators. The construction of this domain was not intended to demonstrate the power of improvising agents, but rather as an example of the use of the Gensim tool in a natural resource management domain. Thus the improving agents are extremely simplistic (mainly because the domain requires very little sophistication), and the domain itself is an extremely simple one relative to any everyday activity. While only a simple and very partial implementation of improvising agents, this work afforded an early opportunity to experiment with some of the computational processes involved in improvisation and laid the groundwork for the implementation described here.

This implementation was performed in Common LISP, using the Gensim simulator described in Chapter 6. The application of Gensim provided a useful element of synergy: the features provided by Gensim provided a natural fit for this application, and the implementation described in this Chapter provided a significant opportunity to test and validate Gensim.

The major difficulty in providing an implementation of a Waffler agent is in the scope of such an implementation. As Chapter 1 has emphasized, building a complete improvising agent would involve solving a great many open problems in artificial intelligence. The implementation described here emphasizes the decision-making components of the improvising agents at the expense of aspects of physical embodiment, such as completely realistic vision or manipulation. The facilities provided by Gensim for obtaining sensory information and manifesting the immediate and temporally extended effects of action are employed to substitute for these features. Other components beyond available technology, such as the parallel working memory required by the architecture, are also simulated within Gensim.

The scope of this implementation is not limited only by the technological resources necessary to construct a complete Waffler agent, however. As Chapters 4 and 5 have illustrated, the collection of background knowledge required for improvisation in any everyday activity is vast and would require an extensive effort to even begin to deal with realistically in the time-span available for the research described in this dissertation. To provide a basis for comparison, the prototype implementation of the Phoenix architecture (described in Sections 3.3, 5.2.4, and 6.3.1), whose domain, while realistic, is comparatively very simplistic compared to the amount of knowledge required to engage in everyday activities, required several people almost two years of full-time development to construct [Howe, 1991]. The CYC project,

implementing the everyday knowledge available to the average six-year-old, estimates a two person-century effort [Guha and Lenat, 1990].

Because of the difficulties of scope on the resource-bounds of this project, the extent of this implementation is necessarily limited. The limited implementation described in this Chapter focuses on narrowing the gap between the model described in Chapter 5 and a complete implementation, within the resource limitations of this project. This is accomplished through an emphasis on the major components of a Waffler agent over the detailed aspects required by a complete agent, and a focus on selected aspects of an everyday activity illustrating the most significant phenomena associated with improvisation. The intent of this partial implementation is primarily to demonstrate the ability of the Waffler architecture to display the phenomena associated with improvisation (as described in Chapter 4) within limited resource bounds. Equally important, however, is the desire to provide an outline for a more extensive implementation and identify both promising avenues of future research and difficulties that must be overcome.

The description of this implementation is divided into three parts: the agent design, the design of the environment the agent inhabits, and a series of examples of agent behaviour within this environment. These are dealt with in the following Sections.

7.1. The Experimental Environment

The domain around which this implementation is centred is one that has been referred to many times in this dissertation; that of making tea in a kitchen. The simulated kitchen defined by Gensim is illustrated in Figure 7-1. The domain is organized using a square grid-structure, with four units in each dimension. Each grid cell is used to represent a large abstract location. For example, the cell labelled *counter1* in Figure 7-1 represents an area of counter space. Many objects may be physically present in this one space, and indeed, it is assumed the agent can move into and around within this space as well. This level of representation is not detailed enough for many applications (e.g. a glass sitting on the counter and an agent standing beside the counter are viewed only as being in the same location), but suffices for the purposes of the examples presented here. Additional information can be recorded for each object indicating facts about its status within the particular abstract location (e.g. on-counter, beside-counter) if this is necessary.

The domain itself contains a varied collection of objects that are commonly found in kitchens, and tend toward those used in making tea due to the

nature of the examples. The precise instances of objects represented in this domain differ slightly in each of the examples shown in this Chapter[100]. However, the main classes of objects are similar.

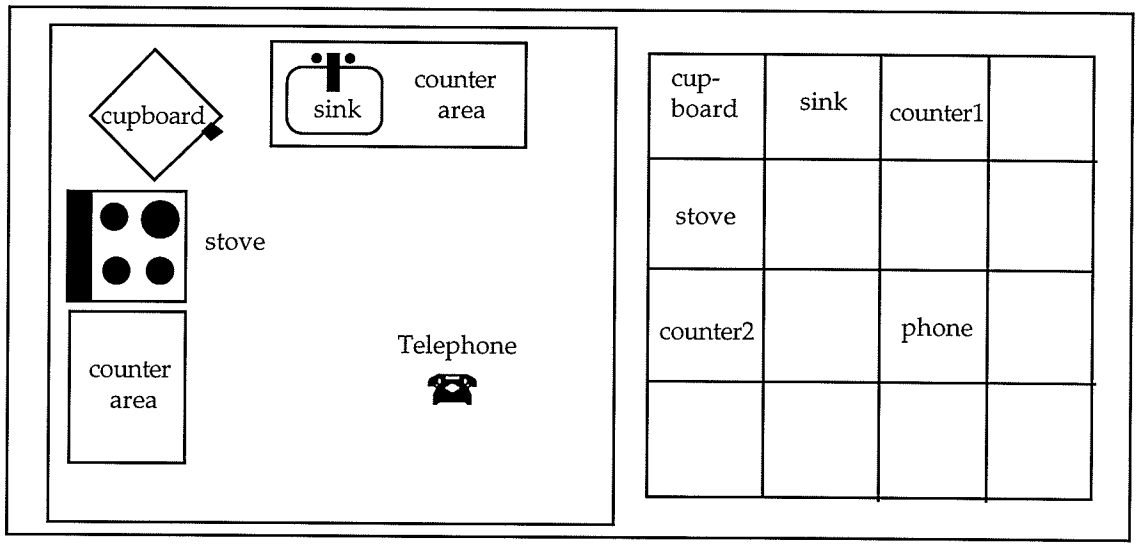

Figure 7-1. Left: Conceptual environment for implemented examples. Right: Environment as a grid-based representation.

The objects in the domain are represented using Gensim's internal frame-based knowledge representation system as described in the previous Chapter. The classes of domain objects defined for each example is illustrated in Figure 7-2. To keep knowledge representation straightforward, multiple inheritance is not used, although it is clearly necessary for a realistic representation of a collection of objects such as this. Pots and cupboards are both examples of containers, as are sinks and stoves. However, to be used realistically each of these objects would also have to be classified according to other criteria: pots as cooking-units, and stoves as appliances, for example. This knowledge base grew as the examples to be described were implemented, and its structure is reflective of this: kettles, for example, could have been grouped as containers or under a new appliance class, but they were originally designated as general

---

100 The reason for these differences is not to remove complexity by having as few redundant objects as possible. Rather, the presence of some objects may affect the behaviour a particular example is trying to illustrate. Consider trying to illustrate how an agent will improvise on its usual routine of using a stove-top kettle for making tea, for example. This is impossible unless the stove-top kettle usually present in the environment is removed.

physical objects and it was never necessary to re-arrange them. Clearly, a more extensive environment would require multiple inheritance, the definition of many new classes of information, and the restructuring of some knowledge defined here.

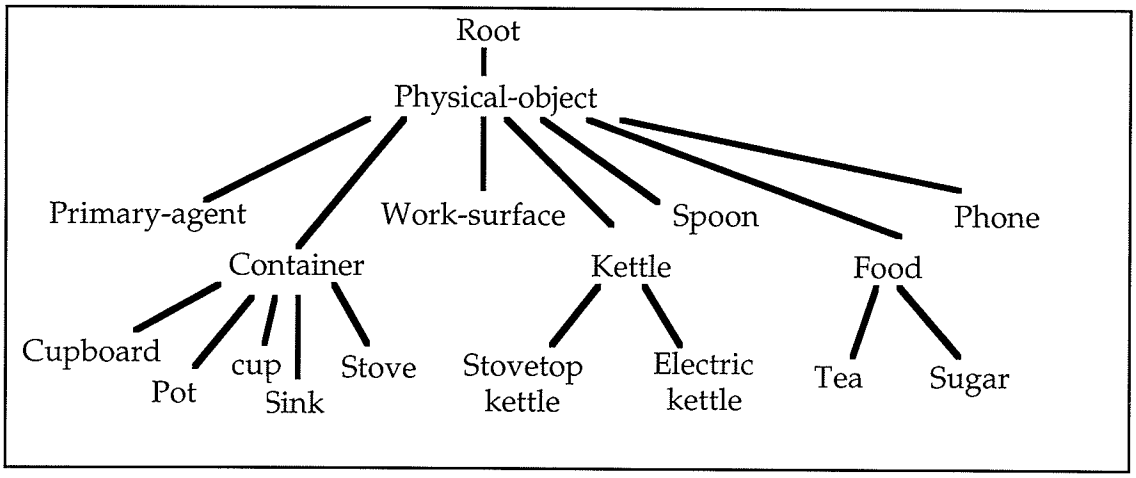

Figure 7-2. List of classes used in example domain.

In addition to multiple inheritance, many other knowledge representation difficulties are ignored for the purposes of this implementation. For example, tea is simply a general concept in this implementation, rather than a set of concepts representing the various forms in which tea may exist (e.g. dry tea leaves, tea as beverage). Transformation (water and tea leaves into tea) is also ignored, although as will shortly be illustrated the Waffler agent does have the possibility to create new concepts in its own knowledge base and link them to existing concepts. Once again, solutions to these and many more knowledge representation problems must be solved before a truly general intelligent agent is possible; however, lacking these solutions cannot and should not prevent experimentation in other areas of intelligent agency.

This implementation makes extensive use of the knowledge separation capabilities provided by Gensim, and because of this it is important to note that the knowledge illustrated in Figure 7-2 is the objective knowledge of the domain maintained by the simulator. In each example, the single agent inhabiting this domain may possess only some of this knowledge, may structure its knowledge in a completely different fashion than that shown in Figure 7-2, and may have additional knowledge as well. In each of the examples described in this Chapter, the agent in all cases possesses identical class information in an identical structuring as that shown in Figure 7-2, but

does not know about all instance information. For example, there may be an electric kettle in the cupboard shown in Figure 7-1, and the agent may not know this fact, but it does know what an electric kettle and a cupboard are, and could recognize an instance of either of these upon seeing one.

7.2. A Partial Implementation of a Waffler Agent

The agent inhabiting the domain described above is a simple implementation of the Waffler architecture described in Chapter 5. This agent contains all the major components described in Section 5.3.1: a long-term memory, a working memory providing immediate realization of the implication of constraints, the ability to trigger the transfer of concepts and intentions from long-term memory to working memory, the ability to deliberate about the utility of potential actions, and perceptual and effectory capabilities.

This agent operates in precisely the manner already described in Chapter 5. The agent's desires and external information trigger the recall of concepts and intentions into working memory, which in turn provide potential actions and constraints to direct the agent's choices for activity. The agent may act at any time upon the recommendations provided by its current knowledge or may wait for more information. The constraints currently active in working memory also form a focus for the agent's perceptual abilities, directing it to information around itself that is pertinent to its activities. All of these are implemented as computational processes within the Gensim simulation tool, except for the agent's perceptual and effectory abilities that are provided using the extensive facilities provided by Gensim for these purposes. The details of the agent itself can be divided into two broad areas: details of the implementation of the constraints, intentions, and other knowledge structures supporting improvisation, and details of the computational processes that apply these structures to produce improvised behaviour. These are described in the following Sections.

7.2.1. Knowledge Structuring

As an autonomous, resource-bounded agent, the Waffler agent operating within the environment described in Section 7.1 has its own limited knowledge of the world it inhabits. Like the simulator's objective knowledge, the agent's knowledge of the world is represented using the knowledge representation facilities provided by Gensim. This Section describes the methods chosen to represent intentions, working memory triggers, constraints, and the other knowledge structures required for improvisation.

The examples of each of these structures used in this Section are drawn from the implemented examples described in Section 7.4.

As mentioned in the previous Section, in all of these implemented examples the agent has as part of its knowledge of the world the complete class knowledge possessed by the simulator (Figure 7-2). However, instance information is limited in the agent's own model of the world, and the agent also has classes of information that are not represented in the objective view of the environment maintained by the simulator. The latter consists of self-knowledge, knowledge of action, and conceptual knowledge over and above that maintained by the simulator.

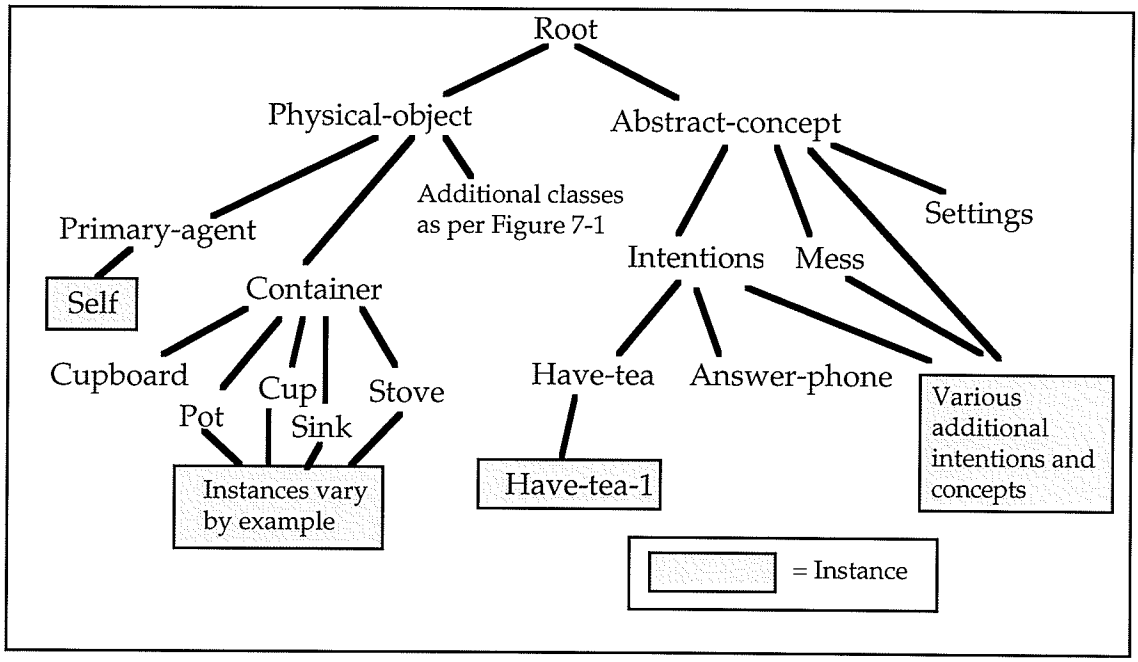

Figure 7-3. General Structure of Agent Knowledge.

Examples of this knowledge are shown in Figure 7-3. In addition to the classes of Figure 7-2 (not all are shown for reasons of space and clarity), the agent has an *intentions* concept that contains knowledge about intentions in general and serves to group the agent's intention knowledge. It also has a *self* instance that serves to maintain self-knowledge, and an *abstract-concept* class that serves to group concepts used in the agent's reasoning processes that do not correspond to any of the classes previously defined. A *mess*, for example, is an abstract concept that would correspond to one or more physical entities maintained by the simulator (such as milk not in any container, or a cup that

is dirty). Roles, such as the general concept of a container for boiling or a stirring utensil, are also examples of abstract concepts.

The Waffler agent in this implementation maintains an extensive model of the world around itself. A world model containing every object the agent has come into contact with is not necessary for improvisation, and it is possible to set up an improvising agent that does not maintain such information. However, the maintenance of such a world model allows the agent to recall immediately where an object that it has seen may be found, rather than requiring repetitive reference to background information indicating where the object is usually stored. The drawback to such a world model is that the agent has the added overhead of distinguishing between when it is recognizing a new object (requiring a new instance in the world model), and when it is looking at something it has already seen. Again, these are issues beyond improvisation in general, and are dealt with in this implementation by assuming that an object seen at a particular location is an existing instance if there is an object of that class known to be at that particular location. That is, if the location and class information maintained for a particular instance in the world model matches that of a perceived object, the existing instance is assumed to represent the perceived object.

Knowledge of concepts, including instances maintained in a world model and abstract concepts used in reasoning, make up much of the agent's knowledge base in this implementation. Several examples of concepts taken from this implementation are shown in Figure 7-4.

The concept in the lower-right hand corner of Figure 7-4, representing the agent's knowledge of the telephone in its world, is typical of most conceptual knowledge in this implementation, containing simple slots and values. The other examples in Figure 7-4 all illustrate various types of knowledge that can be attached to frames. The *kitchen* concept in the lower-left hand corner, for example, contains a trigger associated with an event attached to the phone concept: when the phone rings (the agent is informed of auditory events through the Gensim's signalling facility, described in Section 6.4.3), the telephone concept will be brought into working memory. Triggers may also be defined for visual criteria as well as auditory events such as this one.

The top portion of Figure 7-4 illustrates two more attachments that can be made to basic concepts. The *boiling-container* concept is an example of a role. It represents knowledge about boiling containers, and in this particular case gives a list of classes of items in the agent's knowledge base that can be used to fulfil this role, and the utility with which each fills this role. In this

implementation, utilities are *not* broken up into the separate factors mentioned in Section 5.3.4. Instead, a simple integer representation (where 0 represents the poorest utility, 10 the best) is used for role matches, constraints, and potential actions. This is in part for simplicity in the implementation, and in part to illustrate that extremely complex representations for utility are not always necessary.

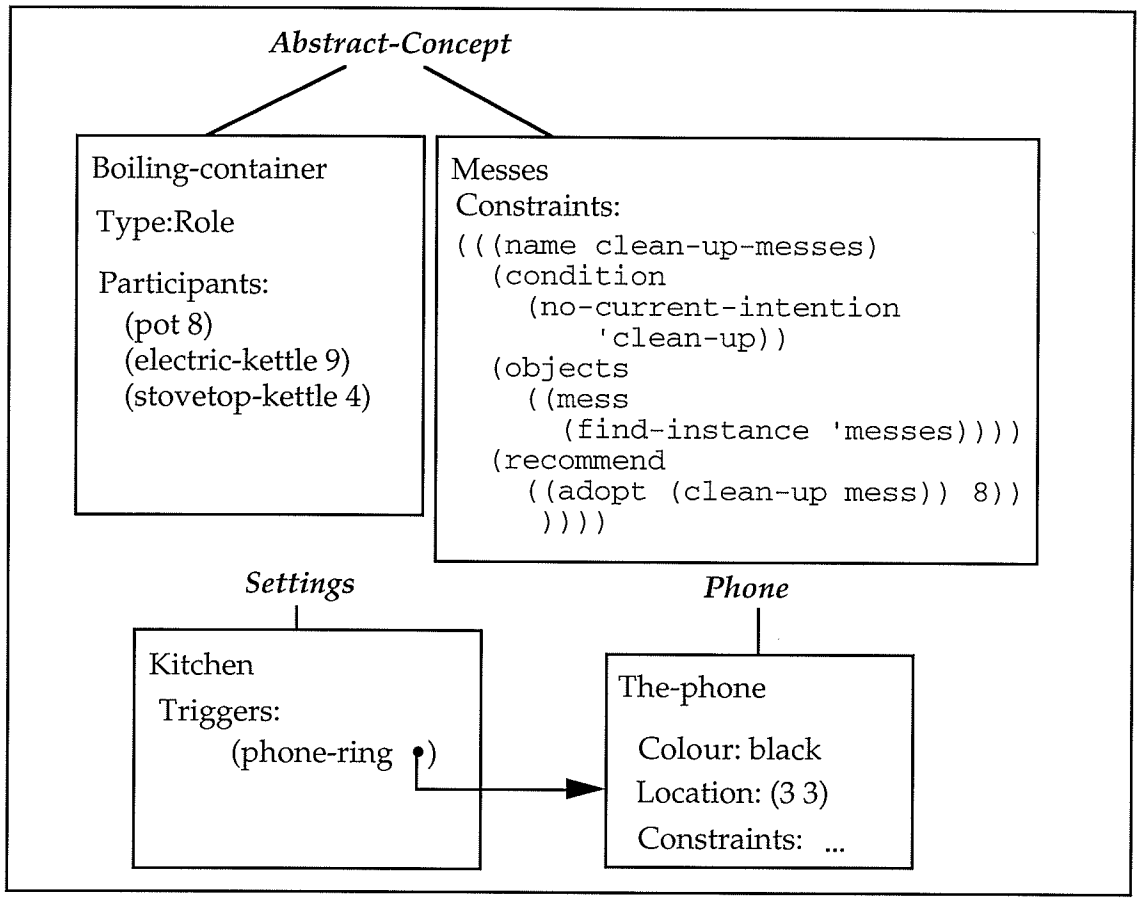

Figure 7-4. Four examples of concept knowledge.

The *messes* concept (this is actually a class, each instance of which is a particular mess the agent knows about) in Figure 7-4 illustrates an attached constraint. For the purposes of this implementation, the implemented components of constraints are a subset of those described in Section 5.1.1. Constraints are represented as a simple association list, with components for a name, condition, recommendation, and objects referred to in the constraint. The recommendation may alter the agent's knowledge base or working memory in some fashion, or may make a recommendation for action. The

constraint shown in Figure 7-4 recommends adopting an intention to clean up a mess, with a utility of 8 (using the representation for utilities described above). A constraint's condition may be any valid LISP expression, and pattern variables may be used in both the recommendation and condition clauses. Bindings for these variables are maintained in the *objects* component of the constraint. These bindings may be inherited from the object the constraint is attached to, or (as in this case) may be created dynamically using another LISP expression. These components collectively make up the bulk of the functionality required for constraints in this architecture. However, the expressiveness of this representation is relatively poor. For example, having constraints' activation conditions as LISP expressions is flexible but requires a great deal of programming to implement complex conditions. A specialized constraint language embedded within Gensim will be required for a more large-scale implementation of a Waffler agent.

Having described the methods by which the agent represents the world around itself and the low-level components of the agent's knowledge representation structures (constraints, triggers), we can now turn to an examination of the backbone of a Waffler agent's knowledge: intentions. As mentioned previously, intentions are organized in their own class within the agent's knowledge base. Each intention itself is also a class, which is instantiated when the agent adopts the intention. Thus the agent may be working on more than one instance of the same activity at a time (e.g. cleaning up two different messes).

Figure 7-5 illustrates two different examples of intentions for having tea, the starting point of each of the implemented examples. As described at the beginning of this Chapter, resource limitations preclude the implementation of the complete activity of making tea. Each of the intentions in Figure 7-5 supports a small subset of the overall tea-making routine and contains only the knowledge necessary for that subset. The intention on the left-hand side of the Figure shows a tea-making intention where the majority of the activity is complete: it remains only to add sugar to the tea and to drink it. The right-hand side of the Figure shows an intention for the same activity at an earlier point in time.

Each intention instance has has an associated *importance*, represented using the same integer scale as utilities. The importance of an intention is derived from both its default importance (stored in the classes from which the intention is instantiated, using an integer scale in the same manner as utilities), and the importance of the higher-level intentions to which this

intention is contributing. The *adopted-from* slot is used to link subsuming intentions for this purpose. Both triggers and constraints (as described above) may also be attached to an intention, allowing the intention to automatically bring associated conceptual knowledge into working memory or make direct restrictions on agent behaviour.

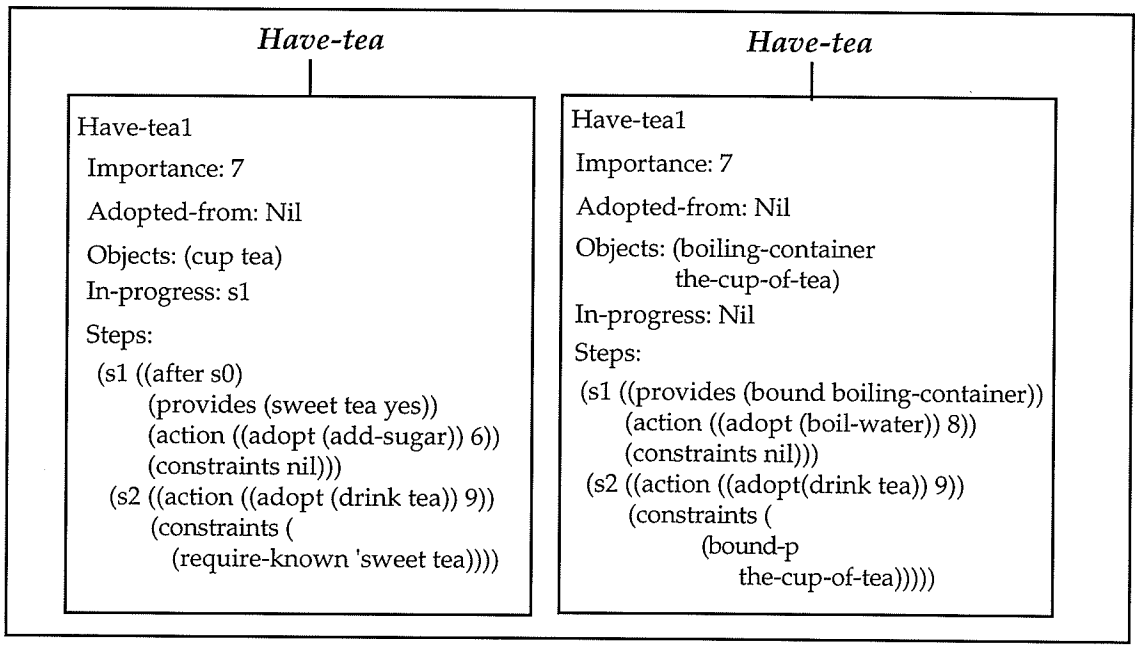

Figure 7-5. Two examples of intentions.

The routine aspects of the agent's intentions may be expressed through a partially or completely ordered set of *steps*. The intentions illustrated in Figure 7-5 both use a series of steps to represent the routine knowledge of the activity. Each step may record the contributions the step makes (in much the same fashion as actions in STRIPS [Fikes and Nilsson, 1971]), may make recommendations for cognitive or physical actions (e.g. step S1 in the left-hand-side intention in Figure 7-5 recommends adopting an intention to *add-sugar* with a utility of 6), and may also have constraints (represented in the same manner as constraints attached to other knowledge structures). The intention on the left-hand side of Figure 7-5 illustrates a strictly-ordered series of steps: Each is explicitly stated to occur *after* some other step. The *in-progress* slot is used to mark the step currently being carried out, and steps are physically removed from the intention instance upon their completion.

The intention on the right-hand side of Figure 7-5 illustrates a more loosely ordered series of constraints: each step has more abstract requirements that

when satisfied make the recommendations of the step available. Like the constraints described previously, intentions have a series of pattern variables whose bindings are maintained in an *objects* slot. Bindings for these pattern variables may be supplied directly to the intention when it is adopted (from subsuming intentions), and steps within the intention may also have bindings for pattern variables as their result. For example, the first step in the intention on the right-hand side of Figure 7-5 illustrates the latter case: deciding to boil water will lead the agent to decide on what type of boiling container to use. This will bind the *boiling-container* object, and this binding will automatically be used wherever the *boiling-container* is used later in the intention.

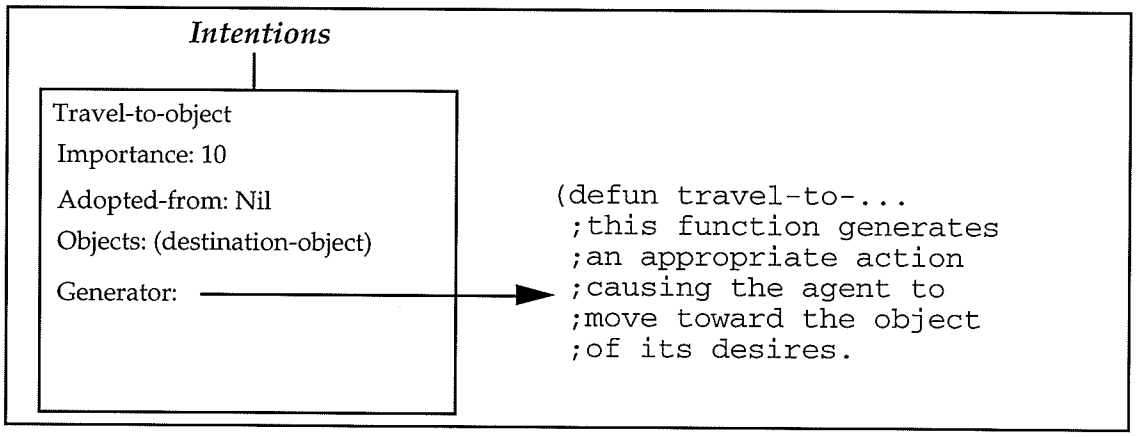

Figure 7-6. Intention with an action generator.

In addition to representing the routine knowledge contained in intentions as a series of steps, an *action generator* may also be used to dynamically generate actions. An action generator replaces the partially-ordered list of steps in the examples shown above with an attached LISP procedure that is responsible for generating an appropriate action when called. An example of this arrangement is illustrated in Figure 7-6, which shows an intention for travelling to a particular object, employing an action generator to produce a potential action to move in the appropriate direction. This particular example is a class rather than an instance: any intention to travel to the location of a particular object will be instantiated from this frame.

Regardless of whether a list of steps or an action generator is used to supply recommendations, it must be stressed that as per Chapter 5, these are *not* actions directly carried out by the agent, but rather recommendations for

action with associated utilities that can be deliberated upon using all the information the agent has at its disposal at a particular point in time.

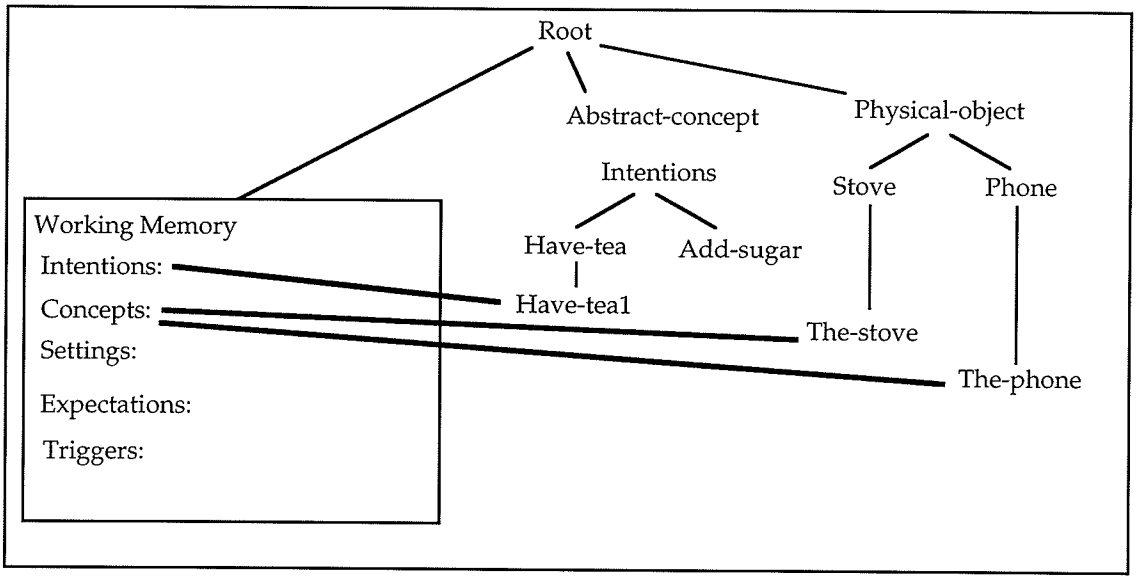

Figure 7-7. Working Memory as a frame structure.

The only major knowledge structure in this implementation that remains to be described is the agent's working memory. While working memory is structurally the most complex component of the Waffler architecture, allowing the effects of constraints contained in it to be realized immediately, it is extremely simple in this implementation. Working memory is simply a frame in the agent's knowledge base holding pointers to the other components in the agent's knowledge base that are currently contained in it, as illustrated in Figure 7-7. The timesharing aspects of Gensim discussed in Chapter 6 are used to handle the parallel constraint processing and triggering mechanisms inherent in working memory. Due to the small size of these examples it was deemed unrealistic to place limits on the number of concepts and intentions that can be maintained in working memory. Given the small number of concepts involved, any limit would have to be extremely artificially small in order to demonstrate concepts and intentions flowing in and out of working memory. Such a small limit would inversely affect the behaviour of the agent and make it extremely forgetful. In lieu of size limits, time limits are used to implement bounded retention in working memory. Each object placed in working memory is timestamped, and any concept not used in the previous two cycles is removed from working memory. Intentions not immediately being acted upon may also be forgotten, with a likelihood in proportion to the importance of the intention.

In general, working memory is the most unrealistic aspect of this implementation, in that the necessary parallelism is entirely simulated. However, as discussed at the beginning of Chapter 6, such simulations are absolutely necessary to allow the development of general approaches without the necessity of solving every underlying technological difficulty beforehand.

7.2.2. Computational Processes

Having described the knowledge structures used in this implementation, the computational processes that operate on these structures can be examined in detail.

The Waffler agent in this implementation is made up of five LISP processes timeshared by Gensim. Because each of these processes is simulating a large amount of parallel information processing in a small time unit, the time limits for each process are unrestricted (As described in Section 6.4.1, Gensim still requires some explicit time limit, but provides the ability to override this time limit for any or all processes).

The first of these processes implements the agent's high-level sensory abilities. Each cycle allows the agent to make a sensory request, and the objects and signals perceived through this sensory focus are provided to the agent at the beginning of the next cycle. This process takes each of these objects and integrates them into the agent's world model. Each time an object is seen, it is matched to the existing instances in the agent's world model to see if it is something the agent knows about. If it is an existing object, any new information obtained is added to the world model and the instance is added to working memory, as are any abstract concepts in which the object participates. The new object (or signal) is also matched to all the triggers in working memory, which may in turn recall new concepts. These triggers would normally operate in parallel, but are implemented in a linear fashion for the purposes of implementation on a serial machine.

If a perceived object is not known to the agent (i.e. is not already in the agent's world model), it is not automatically added to the agent's model of the world. Doing so would assume that the agent cares about each and every object it encounters, and would result in prolific additions and a world model that would eventually (and unrealistically) include every detail of every object in the environment. Instead, a new instance in the world model is created only in very special cases that are particular to the examples involved and which will be discussed in the next Section. In general, objects that are not known are ignored.

The second process in the Waffler agent in this implementation simulates the parallelism inherent in the agent's working memory: it polls each of the intention instances in working memory for its recommendations and attempts to activate all of the constraints in working memory. If an intention in working memory consists of a list of steps, any steps not in some way constrained (e.g. by an unsatisfied constraint or a strict *after* ordering as described in the previous Section) are recommended as potential courses of activity. Each step has a default utility, which is modified by the utility of the overall intention as follows: if the overall intention's utility is 9 or greater, the step's utility is multiplied by a factor of 1.5; 8, a multiplication factor of 1.0; 6 or 7, a multiplication factor of 0.75, 3 to 5, a multiplication factor of 0.5, and less than 3, a multiplication factor of 0.25. If an intention has an action generator, calling the generator to obtain an action becomes a recommendation.

This process also processes each object in working memory, checking for constraints that may make direct recommendations or indirectly alter the agent's working memory. Again, each object is meant to be an independent structure operating in parallel with all the others, and is simulated in linear fashion for the purposes of this implementation. As described in Chapter 5, any links between objects in working memory are also assumed to be realized immediately. The only dynamic links between objects that need to be followed in the implemented examples presented here are *purpose* links. Any object may have an explicit purpose or role associated with it. For example, part of the agent's knowledge of a stovetop kettle may be that it is primarily used as a container for boiling water. The agent may also keep track of this role separately: a *boiling-container* concept in the agent's knowledge base may record general knowledge about this role and offer some potential substitutes. If an alternative for activity uses an object for a specific purpose, and this purpose is in working memory, it may automatically offer further potential alternatives for activity based on substitutions with other objects that are known to fill this role. Thus if the agent is looking for a stovetop kettle for the purpose of boiling water, having this purpose in working memory may generate an alternative to look for an electric kettle[101] instead, automatically making the link between the role used in the original alternative and other objects that may fill this role by virtue of the presence of the purpose in working memory. Other such links may also be immediately

---

101 Or any other alternative maintained by the purpose concept.

realized through working memory as detailed in Chapter 5; however, purpose links are the only such links implemented here.

All of the recommendations made by intentions and constraints in working memory are stored in the agent's working memory frame for access by other processes. If a recommendation is made independently by more than one source (e.g. through an intention and independently through some constraint), the utilities are combined by taking the maximum of all sources and increasing this by one.

The third process finds the most appropriate of these actions by selecting that with the highest utility. It is possible to define constraints on minimal utility levels as described in Section 5.3.4, and so it is entirely possible to have no action reach a sufficient utility to be viewed as a suitable action. In such a situation deliberation can simply ensue over a number of cycles. Additional evidence may accumulate and allow some alternative to move beyond the stipulated limit, or some (internal or external) event may cause the utility constraint to be altered or abandoned, as described in Section 5.3.4.

The fourth process simulates the memory management within working memory that would be required in larger domains. As described in the previous Section, the size of working memory is not restricted; rather this process selectively removes unused components from working memory. When a frame is brought into working memory, the current cycle is recorded. Similarly, each time a concept already in working memory is perceived through the agent's senses and each time a concept or intention is referenced during the agent's course of reasoning, the timestamp on the pointer maintained in working memory is updated to reflect this. This process goes through working memory each cycle, removing concepts that have timestamps two cycles old: effectively, if a concept have not been referenced during the current cycle or the previous one, it is forgotten. This does not in any way affect the agent's long-term memory: it simply removes the pointer from working memory to the structure in long-term memory.

This process also causes intentions to be forgotten. It is nonsensical to forget something on which one is currently working, so any intention on which the agent is currently operating and its associated intentions are immune from this forgetting process. This "current" intention is defined to be the one that contributed the action to be performed on the current cycle. This process takes each intention, searches to find its associated top-level intention, and attempts to forget it if this top-level intention is not the same as that of the current intention.

The issue of forgetting intentions is a contentious one, for forgetting a course of activities is inherently different than simply shifting focus from one concept to another. When a concept is moved out of working memory, it can easily be retrieved: when the object associated with the concept is perceived, for example, or when some intention requiring the object serves as a reminder. Forgetting an intention, however, is more significant. In everyday activities, we sometimes forget intentions only momentarily, more or less as I have described forgetting concepts as above: intentions can simply get moved out of working memory to make room for more locally significant things, and can migrate back in the future. However, in other cases, forgetting is more significant, and the intention is actually destroyed and re-created. We have all at some time walked into a room and completely forgotten what we came for, often having to trace through the last few minutes of activity in detail in to remember. I have not attempted to deal with the latter type of forgetfulness in this implementation, and so forgotten intentions are simply removed from working memory temporarily.

The remaining process is used to make sensory requests and to commit to the agent's chosen course of action. For this implementation, a single sensory request is used. Rather than using the directional *scan* and *look* sensory requests implemented in Gensim, a new *look-around* sensory request was developed, in order to better suit this example and illustrate the ease with which new sensory requests may be developed. *Look-around* is implemented as a LISP function, in the same fashion as *look* and *scan*, and allows the agent to perceive all objects within a one-unit radius around itself.

Actions in a Waffler agent may be divided into two groups: cognitive actions, whose results rest entirely within the agent itself, and physical actions, that may induce internal change but are primarily oriented toward changing the environment around the agent. The latter, as described in Section 6.4.2, are divided into agent-causal and domain-causal components. This final process handles cognitive actions in their entirety, and also handles the internal changes (mainly changes to the agent's world model) made by the agent's physical actions and communicates those actions to Gensim.

This implementation restricts the agent to one action per cycle. At first glance, it would seem that cognitive actions should proceed in parallel with physical actions, the two types obviously not requiring the same physical resources. However, both physical and cognitive actions require time: the former to make physical alterations to the world around the agent, the latter to change the agent's internal state. For the purposes of this implementation, the time required to commit to a cognitive action was viewed as equal to the

time required to handle the agent-causal portions of a physical action, and so only one action per cycle regardless of variety was permitted. This scheme is still unrealistic, however, in that parallel physical actions often occur during everyday activities. This is a much more complicated open problem than that of cognitive vs. physical actions, and involves many issues far outside the boundaries of this project. The use of parallel physical actions during improvisation is an interesting avenue of future research, but is not dealt with here.

The performance of physical actions within Gensim has already been described in Section 6.4.5. The cognitive actions implemented here, however, deserve further explanation. Two basic cognitive actions required of any improvising agent are implemented as part of each of the examples described in the next Section: adopting and abandoning intentions. Each of these is once again a LISP procedure that makes the required alterations to the agent's long-term and working memories to manifest the effects of the cognitive action.

When adopting an intention, the agent supplies the name of the class of the intention as well as desired object bindings (as described in Section 7.2.1, such bindings may be supplied when adopting the intention or may be made during the course of the intention). Information about the intention spawning this new intention is also supplied. When a new intention is adopted, a new instance of the intention is created in the agent's knowledge base. A utility is calculated using the default utility and that supplied when adopting the intention, and the new intention is inserted into the agent's working memory frame, allowing its direct recommendations and constraints to be considered, and its triggers to bring further concepts into working memory.

The parental information supplied when adopting an intention is used when the time comes for an intention to be completed or abandoned. When an intention is complete, it is a simple task to remove it from the agent's working memory frame and destroy the instance representing the intention in the agent's long-term memory. However, any intention may also be part of a larger intention, and upon completion, this may allow new parts of the parent intention to become active. This will be done automatically as a consequence of the constraint-directed representation if the steps in the parent intention are ordered using requirement constraints: the changes provided by the completed intention can satisfy the requirement constraints in some new step. This is not automatic, however, if strict ordering constraints are used: if a requirement of step S2 is that step S1 be complete,

the intention must be informed when this occurs, and thus any intention must communicate its completion to its direct parent. More significantly, any completed intention may also be the only remaining portion of some higher-level intention, and thus any completed intention may recursively trigger the completion of additional intentions. Again, this process requires the parental information supplied when an intention is adopted and stored with the intention itself.

7.3. Implemented Examples

The remainder of this Chapter demonstrates the computational processes and knowledge structure described in previous Sections functioning in three examples of improvised behaviour. Each of these examples was selected because of its clear emphasis on one or more of the basic characteristics of everyday activities described in Chapter 2. All of these examples are set in the kitchen environment of Section 7.1, and have as their basic motive the preparation of a cup of tea. Knowledge particular to each example along with the results of each example are described in the Sections that follow. The actual Gensim output for each of the examples, showing each detailed alteration to the agent's internal knowledge base and the environmental changes of each of the agent's actions, are contained in separate appendices.

7.3.1. Example One

The first example illustrates the agent in the final stages of the making of a cup of tea. The agent has long ago adopted an intention to make tea (this intention is the example in the left-hand side of Figure 7-5). The tea-making intention has been followed to the point where a cup of tea has been made and the agent is adding sugar to the tea to satisfy its preference for sweet tea. The agent has adopted another intention for this purpose, and this intention has also been followed to the point where the tea has been added to the sugar The only remaining recommendations in the intention are to put away the dirty spoon and the sugar bowl used in adding sugar. The environment around the agent looks much as one would expect: a sugar bowl, a spoon, and a cup of tea sit on the counter in the *counter2* area (see Figure 7-1). The agent stands in the same area beside all of these objects, and an event is pre-set to have the telephone ring one cycle into the simulation.

There are several relevant constraints and triggers attached to the agent's knowledge of the world for this example. The *spoon* concept (class) in the agent's knowledge base has an attached constraint that gives a

recommendation to alter the agent's knowledge base to recognize a *mess* when the spoon is dirty. This constraint could also have been attached to a higher-level class (e.g. all physical objects), or to the setting. This is an example of one of the more complex representation issues implemented here: allowing a physical object to participate in multiple abstract concepts. When the agent recognizes that the spoon is dirty, it realizes (through this constraint) that a dirty spoon is an example of a mess, and creates an appropriate instance in the agent's knowledge base to represent this new abstract concept. The agent's mess concept has a constraint that recommends cleaning up a mess when one exists (as previously shown in Figure 7-4). The agent's telephone concept also has an attached constraint that recommends answering the phone when it rings, with a very high utility (10) because of the time-constrained nature of the task.[102] The only setting present in the agent's knowledge base is a kitchen setting, and this setting has a trigger that recalls the phone concept when a telephone ring is heard.

The agent's working memory is initially set to contain the two intentions it has adopted, the kitchen setting, and the agent's concepts of itself, the sugar jar, the spoon, and the cup of tea. The output of this example is contained in Appendix A. This output shows the results of each process described in Section 7.2, cycle by cycle.

At the beginning of the simulation, the agent possesses the two intentions mentioned previously, and these intentions each make a recommendation available in the first cycle. The *have-tea* intention (which has already caused the intention to add sugar to be adopted) has one remaining step, adopting an intention to drink the tea, and this is recommended. The *add-sugar* intention has two remaining steps, putting away the sugar and the sugar jar, and intentions to perform these actions are also recommended.

The concepts in working memory also make contributions in the initial cycle. As described above, the agent's *spoon* concept has a constraint to check for

---

102 This is of course not the only factor that may make answering the telephone important. In many cases, for example, we may expect an important call or know immediately that a call is unimportant. The mechanics of including issues such as this are not difficult: we could simply have a constraint affecting the utility of this alternative. However, as already discussed, there is no end to such complexities in everyday activities, and a limit had to be imposed for the purposes of the resources devoted to this example. Many complexities such as this are ignored in these examples for the reasons described at the beginning of this Chapter.

messes. This constraint is activated, and the agent recognizes and inserts a new instance of a *mess* in its knowledge base and in working memory. This new concept has an attached constraint that recommends an alternative to adopt an intention to clean up the mess.

Each of these recommendations has a utility calculated as described in Section 7.2. The highest utility in this case happens to be the alternative to adopt an intention to clean up the spoon. The agent commits to this action, and a *look-around* sensory request is made (as is done every cycle). This completes the agent's time interval, and the environment is then updated. The agent's action is a cognitive one, and so no changes are made as a result of the agent. As mentioned above, however, an event has been pre-set to cause the telephone to ring, and this occurs during this interval.

The second cycle gives the agent the sensory information corresponding to its request, and the agent perceives the objects around itself, adding a few objects (such as the countertop) to its working memory. The agent also perceives the telephone ring, which in turn triggers recall of the phone concept. The attached constraint is activated, resulting in a highly-rated recommendation to adopt an intention to answer the telephone. The existing intentions make the same recommendations made in the previous cycle, and the newly-adopted intention to clean up the spoon recommends its only step, to travel to the spoon[103]. The recommendation to adopt an intention to answer the phone is by far the most highly ranked, and the agent commits to this action and adopts this intention.

This new intention contains two steps: to travel to an object (the phone) and to talk on the phone once the agent arrives there. In cycle three, travelling to the phone becomes the most significant alternative. This cycle also allows some items to fade from working memory. The agent has moved from acting on its intention to make tea to acting on its intention to have tea (and intentions associated with this). This allows the intentions that go toward tea-making to be potentially forgotten. The original *have-tea* intention is significant enough to avoid being forgotten (a limit of 7 is used), but the *add-sugar* intention is not significant enough and is removed from working memory. This is also the first cycle that could cause items in working memory to exceed their time limits, and several concepts are removed

---

[103] Like the other intentions, the only steps implemented are those that are required for this small subset of the overall activity.

because of this. This leaves the agent with its original *have-tea* intention, an intention to answer the phone, and a new intention to travel to the phone. This intention has a generator (*travel-to*) associated with it that generates actions that move the agent toward the desired object, and these generated actions are more significant than any other recommendations (the only remaining intention is that of having tea, which repeatedly recommends drinking the tea).

The agent requires two cycles to move to the telephone, and as it travels the objects behind it gradually slip from view and are removed from working memory due to time limits (e.g. the spoon, the counter, and the sugar jar). As the agent moves, the phone comes into view (it is already in working memory due to the trigger when the phone rang, but would be inserted through sight if it were not already there). Once the agent arrives at the location of the telephone, the intention to travel to the phone is complete and is removed. Being at this location allows the other step in the *answer-phone* intention, to talk on the phone, to be adopted. Had the agent already been at the phone's location, the constraint requiring this would have been satisfied and the agent would simply have talked on the phone directly - the step to travel to the telephone would never have arisen.

A realistic intention to talk on the telephone would be an extremely complex one, and would involve the agent's knowledge of language and social issues. These are not necessary from the point of view of demonstrating improvisation, and are completely omitted from this implementation. The intention to talk on the phone is thus implemented as an action generator (*chat-on*) that generates a two-cycle delay (two *chat* actions each of which have no physical effects) to represent such activity. After the two *chat* actions, the *talk-on phone* intention is complete and is removed. This in turn completes the *answer-phone* intention, which is also removed.

The agent now has only one intention, its original *have-tea* intention. Like every other cycle in this simulation, it continues to recommend an intention to drink the tea it has made. This intention is adopted, and recommends on the next cycle to travel back to where it knows the cup of tea is sitting. This in turn causes a series of *move* actions that are the inverse of the agent's previous motions. As the agent travels back to the counter area where it was originally positioned, it begins to view the objects in that area (the cup, the spoon and its associated *mess* concept, the sugar jar, and the counter itself) and move them back into working memory. This re-introduces the constraints that were available previously, including the constraint on the mess concept to clean up a recognized mess. This alternative is stronger than

the agent's *drink-tea* intention, and the agent decides to clean up the spoon. The simulation thus ends more or less where it began.

All of this is shown in the detailed output in Appendix A. This example primarily illustrates the Waffler agent's ability to act both directly from its routines and from associated background knowledge. Unless some information indicates otherwise, the agent directly follows the steps outlined in the intentions it has followed many times before. However, alternatives for action also arise from interaction between the agent's intentions and the world around itself, and also arise directly out of associated background knowledge. For example, the telephone ringing triggers a constraint to act toward answering the phone that is completely independent of any other intentions the agent possesses.

This example also illustrates the agent's ability to juggle overlapping activities, one of the common characteristics of everyday activities outlined in Chapter 2. When answering the phone, the agent forgets about its previous activities. In this example, when the agent is finished talking on the phone it simply uses those intentions remaining in its working memory to direct its further activity. This directs the agent back to where it was originally stationed, where visual cues direct it to return to what it was doing independently of its previously-forgotten intentions. This once again illustrates background knowledge and perception substituting for routine. This example would reflect what would happen in the real world if one really did completely forget what one was doing. In reality, however, one might actively analyze one's previous intentions to recall the forgotten intention, something not implemented in this example.

Because the basis for everyday activities lies in personal routine, any reader may see aspects of this portion of ongoing activity that seem unusual when compared to his or her own routine for such activity. Indeed, changes in the agent's knowledge base could make this activity turn out very differently. If the agent were expecting a long call, for example, it might pick up the cup of tea and carry it to the telephone. If the agent had kept the spoon in its hand after stirring and rushed straight to the phone, it would be reminded that the dirty spoon belongs in the sink by virtue of its possession of the spoon. These differences are not in themselves significant. They simply represent differences in routine from that of the agent in this example, and could be handled by altering the agent's routine through appropriate changes in the agent's knowledge base. What *is* important in this example is its illustration of the Waffler agent's ability to handle basic improvisational concepts as described above.

This example also illustrates portions of the architecture that are unrealistic. Upon examination of the output in Appendix A, it can be seen, for example, that the agent's *self* concept is repeatedly recalled and removed from working memory during the course of the activity. This is a result of the policy of forgetting a concept after it has been unused for two cycles. The agent requires no self knowledge during this activity, but knows that it is a physical object in the world and sees itself whenever it looks around. Noting its own presence places its self concept in working memory, which is unused and very soon removed. This is clearly unnecessary but is a result of the dynamics of the various components that make up the agent. This could be handled by treating the agent's self concept differently from others, either keeping it in working memory constantly or recalling it only upon non-visual self-reference. There are many subtleties such as this one, and indeed, many parts of this example in general that could have been implemented differently. As I have already argued however, these issues are not important in the scope of the goals of this example: to provide a simple illustration of a Waffler agent engaged in improvised activity.

7.3.2. Example Two

The second implemented example represents a much shorter temporal period of improvisation, but at the same time shows more sophisticated reasoning on the part of the agent. In this example, the agent is once again making tea, but is at the beginning of the activity. The main intention for this example is that shown on the right-hand side of Figure 7-5. This intention again omits the bulk of the activity, having only two steps: one to adopt an intention to boil water, and another to adopt the intention to drink the tea once it exists.

In this example, several new classes of knowledge are added to the agent knowledge base used in the previous example. Boiling water will involve using a kettle, so the agent is given a general *kettle* concept, along with more specific *electric-kettle* and *stovetop-kettle* concepts. The tea is removed from the cup used in the previous example, and an electric kettle (stored in the cupboard) and a pot (sitting on the stove) are added to the environment. All of these are known to the agent except for the electric kettle[104]. The agent is

---

104 The agent knows about the concept of electric kettles, but does not definitely know there is one in this particular kitchen.

thus not completely familiar with its environment: it knows that a kitchen should have some entity such as a kettle for heating water, but does not know for certain if its present location does. This example also adds one additional abstract concept to the agent's knowledge base. The agent now has a *boiling-container* concept which implements a role: it records knowledge about boiling containers and objects that are suitable for use as boiling containers. In particular, this concept records that pots, electric kettles, and stovetop kettles may be used as boiling containers with utilities of 8, 9, and 4 respectively. The reason for the low utility for a stovetop kettle will be explained shortly.

The agent begins in the same location as that used in the previous example, and has a working memory that initially consists of its intention for having tea and its *self*, *cup*, *spoon*, *sugar-jar*, *stove*, and *pot* concepts. The output of this simulation appears in Appendix B.

As mentioned above, the agent's *have-tea* intention has two steps associated with it: boiling water and drinking tea. There is no tea yet in existence, and so the requirement constraint for that step is not satisfied. Adopting an intention to boil water is recommended, however, and is the only alternative available to the agent. This intention is adopted as the agent's first action, and a *look-around* sensory request is made (as before, this sensory request is repeated on each cycle)

The agent's *boil-water* intention has a constraint on minimum utility associated with it: any step taken in conjunction with this intention must have a minimum utility of 7. Only one step is implemented in this intention: the initial step of choosing how to boil water. This provides a binding within the intention so that the method chosen to boil water (e.g. using a stovetop kettle) is available to other steps and other intentions as the tea-making activity unfolds further.

This step has a preference associated with it that automatically recommends using a stovetop kettle to boil water, reflecting the routine manner in which the agent is used to dealing with this situation. This step also records a purpose of finding a boiling container, causing the *boiling-container* role to be brought into working memory. Had the agent been in its own kitchen, where it is used to using a stovetop kettle and one is readily available, this would immediately proceed. In this case, however, there is no stovetop kettle readily available in this environment. In a more complete implementation, this would be realized by giving the default an initially high value, that would be lowered when the agent is realizes there is no stovetop kettle in the

environment. This could then activate a constraint to find a stovetop kettle to use in the remainder of the *boil-water* intention. In this example, the realization that there is no stovetop kettle is reflected in an initial low utility for the preference (a utility of 4 is used in this intention and in the utility of a stovetop kettle in the agent's concept of the *boiling-container* role). That is, the agent is assumed to possess the knowledge that there is no stovetop kettle, accounting for this low utility factor. The *boil-water* intention thus recommends adopting an intention to find a stovetop kettle with a utility of 4, and has a side-effect of moving the *boiling-container* role into working memory (as mentioned above).

This alternative does not meet the limit on utility and is not considered. There are no other immediate alternatives, and the agent is idle for a cycle while more information is considered. This could potentially lead to either an extensive physical search for a stovetop kettle (by relaxing the restriction on minimal utility) or to an extensive knowledge search to find an alternative to using a stovetop kettle. Neither of these occurs, as a simple solution to the difficulty in which the agent finds itself is readily available. The agent has just moved the *boiling-container* concept into working memory, and has had knowledge of the pot sitting on the stove in working memory since the beginning of the simulation. The presence of the *boiling-container* concept in working memory allows the agent to realize alternatives for boiling containers stored with this concept. as described in Section 7.2.2, this concept stores two additional potential objects to use as boiling containers: electric kettles and pots. These are each put forward as improvised alternatives to the routine preference of using a stovetop kettle. The electric kettle has the highest utility, and the agent adopts an intention to look for an electric kettle. The simulation ends at this point.

The primary purpose of this example is to illustrate active improvisation: using the agent's background knowledge to generate alternatives that may be more appropriate than what the agent's routine recommends. In this example, the agent's routine calls for it to turn immediately to using a a stovetop kettle for boiling water, and the environment around the agent suggests otherwise, causing it to recall the purpose for the stovetop kettle in its routine and to use that to move to find a more appropriate object. Once again, this course of action may differ greatly if the agent has a different routine for making tea. Here I have assumed that the agent's routine is strong, and that it has little experience in making tea using a means other than a stovetop kettle. The agent knows that a pot is sitting right in front if it, and has this concept in working memory. If the agent's pot concept recorded its purpose as a boiling container, this similarity in purpose to the default of a

stovetop kettle would be immediately realized and an appropriate alternative generated. I have deliberately not included this information, assuming that the agent has never before used a pot to boil water for tea and that the sight of a pot alone would not immediately cause the agent to think of this alternative. This forces the agent to recall the role of a boiling container and generate alternatives in the manner described above. That is, the agent does not recognize immediately that the pot can be used as a container for boiling water for making tea, but is reminded of this fact when it recalls the boiling-container role into working memory. If the agent's preference to use a stovetop kettle were even stronger, it would begin to physically search for a stovetop kettle, only pausing to consider other alternatives when a physical search was realized to be fruitless. Once again, these differences are differences in what one thinks of as necessary in a routine for making tea, and have not been omitted due to any implementational difficulty. Each could readily be implemented with only a few modifications to this example.

7.3.3. Example Three

The third example illustrates how a slight modification in the agent's knowledge changes the behaviour exhibited in the previous situation. This example presents the same situation to the agent as the previous example, except that the pot is removed from the environment, a stovetop kettle is placed in the cupboard, and an electric kettle sits upon the stove. The agent's *self* concept also has a high-level constraint attached to it, telling the agent to abandon one intention in favour of the other if it has two intentions that suit the same purpose. The agent begins one step into the example above, having adopted both the basic intention for making tea and that of boiling water. The agent's working memory consists of these two intentions and the agent's *self*, *cup*, *spoon*, *sugar-jar*, and *stove* concepts. The output of this example appears in Appendix C.

The agent begins the simulation positioned just below the *counter2* area shown in Figure 7-1. This is done so that the agent will not immediately be reminded of the presence of the electric kettle on the stove. As before, the agent's *have-tea* intention can make no contributions until some tea exists, and its *boil-water* intention begins by using its preference for a stovetop kettle to generate an alternative to find a stovetop kettle to use for boiling water. The utility in this example is higher to reflect the agent's knowledge that a stovetop kettle exists, and no minimum utility constraint is used in this example. The boiling-container role is brought into working memory as before, but in this case no other concept in working memory matches this

purpose and no new alternatives are generated. The agent thus adopts the intention to find the stovetop kettle.

This new intention (*find-a*) uses an action generator to repeatedly generate actions moving the agent in the direction of the object of its desires. The agent moves once, and becomes close enough to the stove to perceive the electric kettle using the same *look-around* sensory request that has been used in prior examples. This brings the agent's *electric-kettle* concept into working memory, which matches the purpose of the current intention and automatically generates an alternative to the current intention by virtue of its presence in working memory. The new alternative to adopt an intention to use an electric kettle is of higher utility than travelling toward the storage place of the stovetop kettle, and so this intention is adopted. On the next cycle this triggers the aforementioned constraint on the agent's *self* frame, indicating that these two intentions are incompatible and that the less important of the two should be abandoned. This is done, and the agent remains committed to using the electric kettle. Here the simulation ends.

This example illustrates the ability of the Waffler agent to recognize and take advantage of serendipitious occurrences in the environment around itself. The agent is not immediately reminded that there is a convenient alternative to its routine available, and so begins making tea the way it always has. When it sees the electric kettle, it realizes that this is a valid way of accomplishing its desires and improvises on its routine to use the electric kettle. The rest of the routine is not implemented (as in the other examples, for reasons of resource bounds), but the agent could now continue the rest of its tea-making routine using the electric kettle. The agent will still able to rely on the bulk of its routine knowledge for the remainder of the overall activity, and can make use of background knowledge where routine knowledge is unsuitable, or as in this case, where background knowledge indicates more convenient alternatives. This example, like the other two, demonstrates the seamless melding of routine and background knowledge that is characteristic of improvisation. It also illustrates the use of a high-level constraint on the intentions the agent possesses, as opposed to the constraints on individual objects or actions that have been the norm until now.

Like all the other examples, there are many aspects of this that could have been implemented differently. Had the routine been relied upon more strongly, the agent would have continued to move toward the stovetop kettle, completely ignoring the electric kettle. That is, if the agent was more used to relying on this routine, it would seem more convenient to the agent to go all the way to the cupboard to retrieve the stovetop kettle than to use the

electric kettle that presented itself. This could also be explained using social knowledge: in a house other than its own, for example, it would probably be more polite for the agent to use the electric kettle that was available (and which the owner obviously prefers) than to look through the owner's cupboards for a stovetop kettle. Once again, however, these are details in the agent's routine that could easily be added, rather than inadequacies in this implementation.

7.4. Summary

This Chapter has presented three implemented examples of a Waffler agent engaged in improvised behaviour. While none of these examples is complete, and only the first can be considered temporally extensive, they illustrate the architecture presented in Chapter 5 demonstrating its ability to reproduce and deal with the characteristics of improvised behaviour described in Chapter 2. The agent is shown relying on both routine knowledge and the background knowledge behind those routines; dealing with overlapping and competing activities; adopting and abandoning intentions (sometimes to multiple levels); taking advantage of fortuitous circumstances in the environment around itself; and using perception to its advantage. Moreover, it illustrates that a constraint-directed representation is ideally suited to the dynamic nature of improvisation, and demonstrates the power of these constraints in practice.

Having said this, there are still many inadequacies to this implementation. Most significantly, it uses only primitive knowledge representation techniques[105], and it does not implement working memory as the parallel structure required by the Waffler architecture, although it simulates this parallelism in a rudimentary manner. It also does not support parallel physical actions, and does not allow parallel cognitive and physical actions on the same cycle. Working memory also has no physical limitations, and instead relies on a simulated process of forgetting.

Despite all these inadequacies, however, this implementation demonstrates the range of complex behaviour a Waffler agent is capable of exhibiting, and will serve as a useful prototype for more extensive examples of improvised behaviour as well as for experimenting with many of the processes

---

105 That is, primitive with respect to the tremendous knowledge representation requirements of everyday activities.

underlying improvisation, including constraint-directed real-time control and reasoning about function and purpose.

The remaining Chapters in this dissertation draw comparisons to related work and summarize the results of this research.

CHAPTER 8

RELATED WORK

"Men work together", I told him from the heart
"Whether they work together or apart"

– Robert Frost, *The Tuft of Flowers*

8.0. Introduction

In any science, no work exists in isolation. Any scholar stands on the shoulders of those who came before, just as any piece of research has as its foundation the collective experience of the field. Any research will rely on a (usually) fairly compact core of theories as a basis, but will also make use of and have relationships to a great deal of additional work, both completed and ongoing. The research described in this dissertation is no exception to this. There is a core of previous work in which this dissertation has its basis: the theories of classical and universal planning; studies of activity on which these theories in turn are based; the constraint-directed reasoning work of Fox [1981] and others; the studies of everyday activities performed by Norman [1988] and Agre [1988]; and simulation techniques employed by previous AI-based simulation systems.

Rather than treat these works separately, I have endeavoured to make the distinctions between this core of research and my own obvious throughout this dissertation. Chapter 6, for example, extensively describes the relationship between Gensim and previous simulation systems for intelligent agents. Chapters 2, 3, and 4 describe the relationship between improvisation

and other AI-based approaches to activity, both broadly and with respect to some specific research efforts.

The latter is extremely important to re-emphasize, in light of the prominent nature of classical and universal planning. The processes involved in improvisation differ greatly from those used by classical and universal planning, and the approach as a whole differs completely from either of these. The difference between improvisation and classical planning is one of *flexibility*: classical plans are created for a static world and are inherently inflexible. The resources followed during improvisation, on the other hand, serve to guide activity rather than dictate it, and are altered as activity proceeds. The difference between improvisation and universal planning is one of *purpose and direction*. Improvisation indeed involves "continually redeciding what do to" [Agre and Horswill, 1992]; however, improvisation requires a much more flexible basis than a universal plan. Attempting to explain all of behaviour through stimulus-response connections, Jencks and Silver [1972] argue, is accomplishing things through blind chance. A dressmaker, in this behaviourist viewpoint, simply drapes cloth, pulls, pushes, cuts, and manipulates it until it arouses a sense of satisfaction; a poet manipulates words at random, "until a satisfying poem is hit upon quite by chance". An improvising dressmaker or poet, on the other hand begins with experience and a purposeful idea of how to go about the activity in question. The latter approach is creative and purposeful; the former, aimless and random.

In addition to the core mentioned above, there are also important relationships between this research and other projects. The purpose of this Chapter is to briefly describe some of these efforts, acknowledge relationships between these projects and my own, and to emphasize the components of this research that make it distinct. This Chapter examines five projects with significant relationships to work described in this dissertation: blackboard-based planning, partial universal planning, intention-based planning, case-based activity, and constraint-directed multi-agent planning.

8.1. Blackboard-Based Planning

The blackboard architecture [Nii, 1986] is a naturally distributed approach to problem-solving that views the problem-solving process as an incremental and opportunistic. In a blackboard-based approach, a central knowledge store known as a *blackboard* is accessed by a number of specialized entities known as *knowledge sources*, each of which has its own specialized abilities that it

can contribute to the problem-solving process. Knowledge sources contain condition and action components, describing the situation in which the knowledge source is useful and what the knowledge source can contribute, respectively. Knowledge sources become active when their condition criteria are met (through knowledge placed on the blackboard), and contribute their abilities through modification of existing information and the posting of new information on the blackboard. The problem-solving process is opportunistic in that each knowledge source contributes its abilities as soon as it is able, and the solution arises out of the collective effort of the knowledge sources.

Hayes-Roth and Hayes-Roth [1979; Hayes-Roth, 1985] describe a methodology of exploiting the opportunities that arise during the planning process through the use of the blackboard architecture, in an approach to which this research and its precursors [Anderson and Evans, 1991] have been compared.

The research of Hayes-Roth and Hayes-Roth [1979] concerns errand-running in a simulated neighbourhood, and makes use of forty specialized knowledge sources that maintain information about the planning process at various levels of abstraction. Some knowledge sources are domain-specific, and contain knowledge about the area in which the errands are being carried out. For example, a route planning knowledge source may be able to create a plan to get the agent to the locale of one particular errand (e.g. from the starting point to a grocery store). This would then become one partial plan, to be integrated with others (at a future point by some other knowledge source) to create a larger plan to solve all the errands. More sophisticated knowledge sources also exist. For example, a knowledge source might recognize when a resource required by a particular goal is in close proximity to where the agent will find itself at a specific time (e.g. while on the way to the grocery store, the agent will pass by a florist it happens to needs to go to to satisfy another goal). Such a knowledge source, when activated, would propose adding steps to accomplish the second goal (buying flowers) at that point in the plan (merging two partial plans together). Other knowledge sources make more strategic contributions, such as rating one proposal over another. Together, these knowledge sources can opportunistically construct a plan recognizing and exploiting positive interactions between the many goals an agent in such a domain may have.

This work has significant bearing on everyday activities: there are certainly many opportunities that arise during the course of an everyday activity such as errand running, and an architecture designed to approach these must exploit opportunities as this does. This work is inapplicable to everyday activities, however, because the opportunism exploited by the approach is

strictly in the planning process. The plan is never executed, and this "cognitive" model of planning essentially models a human being whom, upon waking up in the morning, plans every step of his or her day before getting out of bed, and then must follow the plan exactly. It is essentially an opportunistic variation on classical planning, with all the difficulties associated with the classical approach.

This architecture also cannot improvise: it compiles knowledge available to it into a plan (albeit in a much better manner than a classical planner), and its specialists then sit idle once the plan is complete. In order to be able to improvise and thus accomplish everyday activities, these specialists must be able to contribute their knowledge as the plan unfolds. Although the stored intentions on which the Waffler architecture relies are used as guides to particular activities, there are no specialists in the sense of the term used in the Hayes-Roth's approach. The act of realizing, for example, that the agent is close to a resource needed for a particular goal when carrying out some other course of activity is made possible by knowledge of the activity itself, not a specialized knowledge source. The need to be in a particular place keeps the concept of that place in working memory, and seeing it (or recalling the concept in some other fashion, such as realizing it is on the route one is taking) triggers the opportunity (in the form of a constraint). Specialized knowledge is distributed in part with the activity it pertains to, or to some class of activities, and becomes active as long as an intention it is associated with is in memory. Such knowledge also exists in the form of a set of distributed concepts that are associated with intentions. Opportunistic control in the Waffler architecture is a function of a distributed constraint-directed representation, rather than that of a collection of knowledge sources.

By making use of both compiled knowledge and the knowledge from which those compilations are made, the Waffler architecture has an ability to react to opportunities equal to that of the Hayes-Roths' approach. The Waffler architecture can react opportunistically not only to the realization that its current situation can assist in accomplishing some other goal, but also to dynamic changes in the domain, something beyond the capabilities of the Hayes-Roth's approach. Further, my approach encompasses regulating the amount of time spent planning (in the sense of exploring background knowledge to find an appropriate response), a problem that was not considered by the Hayes-Roths' because of the lack of any execution component in their model.

Despite these differences in processing, there is one physical characteristic of both architectures which appears similar and invites comparison: the active

nature of both constraints and knowledge sources. A constraint in the Waffler architecture can in some sense be compared to a blackboard knowledge source: both are applicable in given situations and make some contribution when activated. Beyond this basic similarity, however, the two are very different. Knowledge sources are course-grained, and may involve complex processing devoted to some specialization. Constraints, on the other hand, are fine-grained pieces of active knowledge that make simple contributions to the agent's current potential actions or to its knowledge base.

More significantly, the active nature of the constraints and triggers in a Waffler agent's working memory is a result of their presence in working memory and is in fact a characteristic of working memory rather than these structures themselves. In a blackboard approach, all of the knowledge sources are available for activation at any point in time. As already mentioned, the number of active constraints that would be required to deal with any everyday activity in its entirety would be enormous, and thus a blackboard approach could never suffice.

8.2. IRMA: An Intention-Based Architecture for Rational Activity

Another work that bears some similarity to the Waffler architecture is the intention-based planning architecture of Bratman et al. [1988]. This architecture, unnamed in [Bratman et al., 1988], later became known as *IRMA*, and will be referred to as such in this account.

IRMA is based on the integration of means-end reasoning (goal-directed planning) with the rational weighing of competing alternatives under resource-bounds. The architecture is intended to mediate between the need to commit to ongoing plans and the need to re-examine those plans in light of changes in the environment. An IRMA agent's current plans act as constraints on future behaviour, in the sense that commitment to a given plan will cause the agent to avoid performing actions that will interfere with or invalidate the plan. The IRMA architecture consists of four core components: a *means-end reasoner*, an *opportunity analyzer*, a *filtration process*, and a *deliberation process*.

At any given time, an agent will have a number of structurally partial (skeletal) plans that it will be following. The means-end reasoner takes those plans and generates goal-directed potential actions. The agent also has a set of beliefs about the world, that are modified through perception of changes in the outside world. Changes in beliefs are input to the opportunity analyzer, which separates relevant changes from irrelevant ones. The options (from

both the opportunity analyzer and the means-end reasoner) are then subject to filtering. The *compatibility filter* (one component of the filtration process) checks the compatibility of options with existing plans. Those that are compatible are *surviving options*.

Surviving options are passed to the deliberation process, where competing options are weighed against each other. These produce more intentions which are incorporated into the agent's plans. Filtration and deliberation are really the same function at two levels - the agent eliminates some options and form a small set of the most appropriate, and then uses this as a basis for further deliberation.

The filtration process is the most important part of the architecture, mediating between the stability that plans provide and the revokability of plans. In addition to the compatibility filter mentioned above, a *filter override mechanism* (operating in parallel with the compatibility filter) encodes the agent's sensitivity to problems and opportunities in the environment. An option "weeded out" by the compatibility filter may still be subject to deliberation if it triggers a filter override (that is, if it is incompatible with the current planning direction but is still an important option). When a proposed but incompatible option triggers a filter override, the agent is being *cautious* [Bratman et al., 1988]. On the other hand, when a proposed but incompatible option does not trigger a filter override, the agent is being *bold*.

This override mechanism must be carefully designed for a specific domain to provide the right degree of sensitivity. If the agent is willing to reconsider plans in response to every unanticipated event, plans will not serve to limit options for deliberation. However, if the agent is not sensitive enough, it will fail to react to significant deviations from its expectations.

The most significant similarity between this work and my own is in the use of the term *intention*. The sense of intentions used in IRMA is that of Bratman [1987], described in Chapter 4. An IRMA agent's intentions are plans that are partially carried out or will be carried out at a future time. They perform some of the general functions of intentions in my own work, in that they serve as constraints on agent's future behaviour. However, both the philosophical and physical nature of intentions in these two works are very different. The intentions of Bratman et al. are simply skeletal plans, of the same variety used by Phoenix [Cohen et al., 1989], as described in Section 6.3.1. Bratman's intentions serve as constraints by virtue of commitment: that is, the effort the agent has put into constructing the plan and which would be

wasted if the plan were abandoned or compromised. In the Waffler architecture on the other hand, intentions serve to constrain future behaviour by virtue of the fact that they represent the routine knowledge that has served the agent in the past during the course of the activity: not because of the effort spent constructing them. The constraints that intentions place on future behaviour are specific constraints concerning the activity as a whole, the interaction between that activity and others (or between the activity and objects in the environment) or concerning objects or low-level actions that form a part of the activity. That is, the constraining action is a *specific feature of the knowledge structure of intentions and their interaction with the environment*, not a vague side-effect of their existence. Intentions in the Waffler architecture are also very different in structure than the intentions used in IRMA. Again, Bratman's intentions are skeletal plans: their constraining action lies in the nature of the structure itself, and they represent only compiled knowledge. A Waffler agent's intentions are active structures, representing the general structure of the activity in which the agent is participating to a much greater extent than the intentions used in IRMA. The fact that the background knowledge behind the compiled parts of an intention is considered part of the intention itself makes the intentions of the Waffler architecture more flexible and more widely-applicable than that of Bratman et al. [1988].

The fact that background knowledge holds no place in the intentions used in IRMA makes both the concept of intentions and the architecture itself vague. Background information is certainly necessary in the architecture in order to provide filtering and filter overrides, but the nature of this knowledge and the manner in which it is to be represented and accessed are ignored. The original architecture is described as "a sketch", and it is precisely that. The nature of background knowledge, filtration, the analysis of opportunities, and filter overrides are all treated as implementational details, to be (presumably) easily conceptualized for a specific domain or application. Unfortunately, each of these components is much more than an implementational detail. In order to apply this architecture to everyday activities, much more concise specifications of each of these would be necessary. Specifying any of them will change the conceptual architecture greatly (and would be a large research project in itself).

Two simple implementations of an IRMA agent has been performed using the Tileworld: the first to demonstrate the performance of the Tileworld simulator [Pollack and Ringuette, 1990]; the second as a basis for a series of experiments on agent commitment to goals [Kinney and Georgeff, 1989]. Recall from Section 6.3.2, however, that the Tileworld is an extremely simple

tile-shifting environment, and the use of plans, the dynamic nature of the environment, and the amount of information an agent must deal with in no way reflects the complexity of the real world. Pollack and Ringuette, for example, describe a simple formula that can be used to reliably rate a plan to fulfil a potential goal, based on the score value and distance of the hole. No such simple function exists for everyday activities. Likewise, the implementation of filtration, filter overrides, and the relationship between the two is trivial when compared to everyday activities. Pollack and Ringuette [1990, p. 185] dismiss this important aspect of the architecture by reducing these problems to a single statement:

> The crucial task for the designer of an IRMA-agent is to construct a filter override mechanism so that it embodies the right degree of sensitivity to the problems and opportunities of the agent's environment.

This is much easier said than done. It is my contention that this architecture would be impossible to use for the purpose of accomplishing everyday activities, because of the complexities mentioned above. The most difficult aspects of accomplishing everyday activities are tied up in precisely those parts of the IRMA architecture that are left open: in balancing the use of plans (compiled knowledge) with other information.

As a sketch, the IRMA architecture has provided an illustration as to how resource-bounded reasoning can be carried out in a common sense fashion. Pollack and Ringuette [1990] describe the distinct contribution of IRMA as using means-ends reasoning to produce options that are then the subject of deliberation. The Waffler architecture is certainly inspired by this principle. However, the Waffler architecture provides solutions for those aspects of IRMA that are left open, through the application of constraint-directed reasoning. Constraints can represent the structure inherent in any everyday activity and the environment in which it is performed, as well as interactions among activities and between an activity and the environment. Constraints can also be used to control resource-bounded reasoning. The latter point is another major difference between IRMA and the Waffler architecture. I have argued and illustrated that constraints themselves can serve the purpose of controlling resource-bounded reasoning, and the integration of various activities in a rational manner. No explicit filtering step such as that proposed by Bratman et al. [1988] is necessary: it is provided by the choice of representation.

8.3. Partial Universal Planning

In Section 3.2, Universal planning was criticised for its very assumption of universality: the fact that no complete mapping of situations to appropriate actions is possible in any significantly complex domain. Dean et al. [1993] attempt to overcome this difficulty by creating partial universal plans. The system initially constructs an universal plan that is incomplete, by virtue of the limited knowledge it has of the domain and and the restricted time it has to construct the plan. This limited universal plan will contain many unfilled states: that is, states that have no action mapping to them because they are unknown to the agent or because an appropriate action to be taken when in that state was not generated in the limited amount of time given to construct the plan. These *external* states have default actions associated with them, allowing the agent to move outside of the universal plan in a limited manner. These default actions may include extending the universal plan to cope with the unrecognized situation.

Partial universal plans are extended as the agent is given more information and more time to plan. This extension follows a two stage process: new states are added to the restricted universal plan, and the situation-to-action mapping is then re-computed, resulting in new and presumably better action mappings. This two-stage process allows a universal plan to be improved upon both by extending it to recognize new states and by optimizing its performance through re-computing the situation-action mapping to include new information. The construction of limited universal plans relies on a rating of the *reward* derived by being in a particular state, and known probabilities that an action will cause a transition from one particular state to another. Dean et al. [1993] illustrate this approach by showing a partial universal plan being generated for a robot motion planning domain and extended as the agent views more of the world around itself.

In a limited sense, this is improvisation: the system does its best with the information it has, and given more time to make a decision, uses more information to (presumably) make better decisions. This architecture is very promising, in that it is the first serious effort to examine the use of incomplete universal plans, as well as the construction of universal plans. Previous efforts in universal planning (e.g. [Agre, 1988; Chapman, 1990]) have relied on pre-constructed, hand-tuned universal plans, giving little or no indication of how these would evolve through experience.

While Dean's architecture does perform improvisation in the sense that it makes use of the information it has available to it, this is only one of the

colloquial interpretations of improvisation discussed in Section 4.1. The approach of Dean et al. does not, however, satisfy the definition of improvisation adopted in this dissertation. Dean's architecture suffers from the same basic problem as the classical, interleaving, and universal planning architectures described in Chapter 3: it relies completely on compiled knowledge, and disregards the much more basic knowledge from which those compiliations are derived. Dean's architecture performs better than the other alternatives, because it has the ability to extend its universal plan when something goes wrong. However, this is still interpreted in terms of a plan-then-execute strategy: when a state outside the current partial universal plan is arrived at, the agent can default to a predefined action or call its planning routines to extend the partial universal plan. The Waffler architecture, on the other hand, performs much more naturally, following a mixture of compiled and loosely-associated information and relying on compiled knowledge to the degree that that the situation allows.

Dean's architecture also suffers from technical problems that make it unsuitable for application to everyday activities. In developing their approach, Dean et al. rely on having a known deadline before starting to construct a universal plan for the environment in which the agent finds itself. As described in Chapters 2 and 4, this is completely inappropriate for everyday activities. Rarely do we work with specific deadlines in everyday activities; instead, the importance of the activity with regard to the world around the agent, changing moment by moment, influences the agent to act in a specific goal-directed way (effectively, a soft deadline that could occur at any moment).

In a more general sense, the entire approach of incremental universal plan construction is unsuitable for everyday activities. Chapter 2 has characterised everyday activities as those that are stable and well-known to the agent involved. However, new information arrives quickly when performing these activities, and everyday activities interact heavily with one another, making the very concept of constructing a universal plan on a moment-by-moment basis highly questionable: the plan would simply change too significantly, too quickly, as the agent focused on various portions of a single activity, integrated it with others, and considered its future activities. Moreover, the whole point of improvisation is to use knowledge of an activity, its setting, and knowledge of action in general in order to perform that activity in a specific, dynamic situation. Thus I can use my knowledge of making tea to make tea at my neighbour's house, overcoming the differences through general knowledge of the tea-making process. Universal plans, on the other hand are specific, with each situation specifying a precise set of

criteria for a given action to be used. Thus, rather than making small alterations to a fairly static universal plan, the agent would be essentially constructing plans *de novo* much of the time.

By relying on universal plan construction, Dean's architecture also fails to take advantage of the structure available in everyday activities (Chapters 2 and 4): it would be a much more useful approach to recall as large a portion of plan knowledge as possible from a knowledge base representing previous experience, as is done in the Waffler architecture. Universal plans do not, however, support such phenomena. Finally, Dean's architecture is to universal planning as interleaving architectures are to classical planning: it is an improvement on a basic approach, meant to extend the approach to a form of activity for which it was not designed. Like interleaving systems, Dean's architecture suffers from problems inherited from its parent. Just as in universal planning, for example, Dean's architecture can look only one step ahead at a time, and must rely solely on current stimuli for action decisions.

Like the approaches discussed in Chapter 3, Dean's approach is unsuitable for everyday activities because the methodology on which it is based was designed for one small part of the spectrum of activity, with entirely different requirements from those of everyday activities. However, the approach is well-suited to other areas. This work was not designed specifically for everyday activities, and indeed, the major focus of Dean's work is completely different from that described in this dissertation: the time-dependent generation of partial universal plans, as opposed to the flexible application of plan knowledge. The criticism in this Section is presented to serve only to differentiate Dean's approach from that presented in this dissertation, and to show its unsuitability to everyday activities.

8.4. Case-Based Activity

The research presented in this dissertation also bears some similarity to recent work on case-based reasoning, most notably the *Trucker* and *Runner* projects [Hammond, 1989b; Hammond et al., 1990]. These projects attempt to extend techniques of case-based planning [Hammond, 1989a], where improved plans are constructed based on mistakes previously made by the planner, to more dynamic realms.

The Trucker and Runner systems deal with essentially the same kind of problems in domains of different complexity. The Trucker system [Hammond, 1989b; Hammond et al., 1990] involves planning in a package courier domain. A number of delivery vehicles inhabit a simulated city, and

a series of orders (for pick-up and delivery of packages) arrive dynamically. Trucker makes use of a map to find pick-up and delivery points, and stores routes it has travelled in a case memory, so that routes can be re-used in later travels. Trucker also has a means of recognizing opportunities: if a pick-up is scheduled at a later time, and a vehicle is passing the pick-up site, the system can recognize and take advantage of this by including the route to deliver the new package in its current route if it is not too expensive to do so. When it includes a secondary goal in this manner, the new route plan is stored indexed on both locales, so that should a vehicle be going to the same initial goal, it can check for a possible pick-up at the same passing location again. That is, the agent can recognize the opportunity in future.

Runner [Hammond, 1989b; Hammond et al., 1990] implements a more sophisticated errand-running domain, based on the domain of the Hayes-Roth's opportunistic planner (Section 8.1). Runner is only a single-agent system, but because of its broader domain, the system demonstrates a much wider range of activity than does Trucker. Runner follows stored plans to perform everyday activities such as making coffee. These plans consist of nodes representing actions, each of which has a series of conditions for the action to be performed (e.g. a *fill-pot* action has conditions such as being near the coffee pot and knowing that the coffee pot is empty). The system supports the recognition of opportunities in the same manner as Trucker, only in a more sophisticated domain. If a Runner agent desires orange juice, for example, goes to the refrigerator to get the orange juice, and then finds none, the original goal is suspended and indexed in the agent's memory with the conditions that will allow its future satisfaction. An agent is then reminded of the goal when those conditions arise. Hammond relies on indexing only by specific conditions on a blocked activity, in order for the agent not to be reminded of a goal too often (e.g. indexing the orange juice goal by being at a store, rather than having the money to buy orange juice) [Hammond, 1989b].

In its use of stored plans and indexing, these systems would seem to be similar to Hammond's earlier work in case-based planning [Hammond, 1989a] described in Section 3.4. However, these systems carry out plans in a much more sophisticated manner. Both systems make use of simulated environments, using a parser to recognize features in the environment and activate larger concepts (e.g. seeing a lighted sign, a building with windows, and cars in a parking lot would trigger the concept of a store). Individual actions from within stored plans are activated by recognizing their specific conditions via perception, and these actions are made available for execution by *permissions* passed down from the plan level. Differences in the specific application of an action (e.g. *grasping* in one situation vs. another) are

assumed to be associated with features in the environment, rather than the plan. That is, actions are specialized by features in the environment, allowing the world around the agent to determine the specific method used (after Firby [1987]). All activity is motivated through the violation of policies. Policies (as defined by McDermott [1978] and described in Section 5.1.2) are defined as statements about the ongoing goals of the agent (e.g. *always have money*).

The most obvious similarities between this system and the Waffler architecture are in the ability to allow goals to interact with one another (as illustrated by the first implemented example described in Section 7.3.1 and Appendix A), and in the use of policies as constraints on behaviour. These aspects will be dealt with first.

The Waffler architecture essentially duplicates the abilities of Trucker and Runner to block goals and recall them later based on given conditions. If an improvising agent wants milk for its tea, for example, and goes to the refrigerator and finds none, that fact can be stored with the agent's milk concept, to be recalled if the agent happens to come across milk (a laissez-faire approach), or more actively, by associated the need for milk with a store, thereby reminding the agent to buy milk when it happens to recall the store concept[106]. The Waffler architecture goes beyond Hammond's work however, by giving the agent the ability (through constraints) to consider going out immediately and buying milk. Runner can link suspended goals to the situations that are advantageous to their satisfaction, but has no real ability to weigh one alternative against another. In the case of the orange juice example mentioned above, Runner simply assumes the agent will eventually stumble upon a store.

To be fair, the intent of the Waffler architecture and Hammond's work are very different: Hammond is concerned with learning information during the course of planning and carrying out plans. As such, the methods used to create links between blocked goals and the conditions for their solution are much more sophisticated in Hammond's work than in the Waffler architecture. The ability to link a blocked goal (e.g. no milk) to its satisfaction conditions (the agent's store concept) is provided only at a basic level in a

---

106 It should be noted, however, that this ability is not included in the implementation described in Chapter 7. The first implemented example (Section 7.3.1), which illustrates an agent being interrupted by a telephone call while making tea, has the agent answer the call but does not completely forget its initial activity. This was simply a choice made at the outset of the example; this ability could be implemented with little additional effort.

Waffler agent: the link is made by virtue of the agent being told that this is appropriate in its knowledge base. That is, because learning is not a focus of my work, the responsibility for deciding the propriety of such a link is left with the designer of the knowledge base. Hammond, on the other hand, provides a theory of how such links should be made [Hammond, 1989b; Hammond et al., 1990]. That is, when confronted with a number of possible criteria for the satisfaction of a blocked goal, Hammond provides criteria that allow the agent to separate conditions that make useful links from inappropriate conditions. Trucker and Runner create their own links based on these criteria.

The implementation of policies and the view of constraints in general embodied in the Runner and Trucker systems is also very different than that embodied in the Waffler architecture. Hammond views an agent has having a collection of policies (essentially high-level constraints on the agent's behaviour) whose violation generates goals. Violation of these policies is the sole means by which activity (the adoption and execution of plans) is initiated. I have already argued, on the other hand, that the violation of constraints occupies a comparatively minor role in everyday activities, and must do so to be even remotely tractable. Realistically, having policies such as *always have coffee in the morning* is both over-generalizing (even habits are not laws) and unrealistic (any agent would be continually sifting through thousands upon thousands of irrelevant policies). The Waffler architecture, on the other hand, uses policy constraints as only one small portion of a much larger collection of constraints. Activity is not dictated by policy but is achieved through electing to follow the constraints from many sources. The only relevant violations to constraints are violations of an agent's expectations, since these form a basis of future activity.

More significantly than either of the above criticisms is the primitive nature of deliberation and action selection in Runner and Trucker. Hammond et al. [1990] state that action conflicts (e.g. when two actions are active, such as filling the coffee pot vs. putting coffee in the filter) can be handled through special mediation actions, which act as rules that will determine the ordering in plan actions. Considering this from the point of view of complexity, it seems difficult to conceive of such rules being able to arbitrate between actions contributing toward totally different goals, as I have illustrated in the

examples in Chapter 7[107]. There are simply too many possible actions for this approach to be feasible. Indeed, a complete collection of such arbitration rules would form a universal planner, the difficulties of which have already been discussed (Section 3.2).

In the Waffler architecture, the constraints associated with actions and intentions will bias the agent toward one action or another, or if the interaction is an extremely common one (such as the above example), a preference may be available. Preference constraints in a Waffler agent serve essentially the same purpose as Hammond's mediation actions, but are not relied upon completely. Indeed, in the majority of cases an explicit preference will not be available. The improvisational approach hinges on the ability to reason *behind* such preferences (and other forms of compiled knowledge). Hammond's mediation actions are compiled knowledge, and the Runner system shows no use whatsoever of knowledge outside of a plan. Improvisation can move beyond this compiled knowledge when necessary, resulting in a much more robust approach to activity. In no way could the coffee plan shown in [Hammond et al., 1990] be adapted by the system to work in a different environment (since activation of an action such as filling the coffee pot depends on it being in plain view, which might not be the case in a less-than-familiar setting). In point of fact, Hammond assumes that each plan used in Runner will fit a specific case, and that the agent will have one plan for each particular setting. I have already argued (Chapters 2 and 3) that this approach is untenable. Hammond et al. [1990] also do not give any indication as to how a blocked plan can be recovered from save by suspending the plan and indexing it to the conditions facilitating its resumption.

Neither Trucker nor Runner has the ability to improvise as the term is defined in Chapter 4. The ability to move beyond compiled knowledge is thus the most important distinction between Hammond's work and my own. Actions in Runner and Trucker have both goal-directed (via permissions) and adaptive (via activation) aspects; however the adaptive behaviour of the approach is extremely primitive when compared to that used in the Waffler architecture. [Hammond et al., 1990] describe an extensive example of coffee-making using Runner, showing the system following the plan by activating actions using perceptual information and giving them permission to be

---

107 Hammond et al., 1990] illustrate no such examples, nor what occurs when a plan goes awry (outside of blocking the goal and resuming it later). Indeed, Hammond et al. do not show how *any* knowledge outside of the plan can be introduced to the deliberation process.

carried out one by one via the agent's plan. However, the idea of mediation between action through anything other than brute preferences is non-existent. If there is no preference between two actions with both activation and permission, the system is baffled. Even if a preference exists, that preference serves as a law rather than a recommendation. This is essentially the same problem associated with classical planning systems: the use of compiled knowledge without maintenance of the knowledge from which those compilations are made.

Hammond also argues that one of the benefits of his approach is a unified representation. The only really unified aspect of Runner and Trucker is the use of activation and feature recognition in both perception and action activation. The architecture presented in this dissertation, on the other hand, makes use of a truly unified representation: domain knowledge, knowledge of action, knowledge of the relationship between the agent and its environment, and control knowledge are all represented and managed using constraint-directed reasoning.

There are, of course, also aspects of the Trucker and Runner systems that are better than the Waffler architecture. The unification of recognition in the environment and activation of plan elements is more sophisticated, because these systems work with a more sophisticated model of perception that that provided by Gensim. This is not, in my opinion, a significant difference. Much more significantly, Trucker and Runner embody theories of learning that are not encompassed by the Waffler architecture. Trucker, for example has a methodology for storing new routes, although it does not seem to discriminate between useful knowledge and knowledge that will have little effect on its performance. Indeed, one would think that the brute force recording of previous pick-up and delivery locations would eventually degrade performance as a result of the presence of so many redundant pieces of knowledge. The learning of intentions is beyond the scope of this dissertation and is, I believe, a much more difficult problem than implementing learning in Runner or Trucker, because the memory structures used in intentions are much more dense, and the boundaries between intentions are indistinct.

To summarize, the Waffler architecture possesses abilities to recognize opportunities and recall previous goals comparable to those of Runner and Trucker. The constraint-directed approach used in improvisation, however, provides unifying aspects of reasoning and control not found in Hammond's work, and my architecture has the ability to move beyond compiled knowledge to the background knowledge from which those compilations

been obtained. Improvisation as I have defined it in Chapter 4 is beyond the scope of Hammond's systems.

8.5. Constraint-Directed Multi-Agent Planning

While the previous Sections of this Chapter have all described similarities between the Waffler architecture and other architectures for integrating reactive and deliberative reasoning, the Waffler architecture also has a significant relationship to work done in a completely different area, that of multi-agent planning.

Many practical planning problems resist solution by a single agent, because the problems are spatially distributed, because the volume of work will exceed the resources of any single agent, or because the problems demand specialized knowledge or abilities that are unreasonable to expect from any single agent. Such problems require the development of agent architectures that can not only cope with the behaviour of other agents in the world around themselves (as any intelligent agent must), but can actively cooperate with those agents to achieve goals that are beyond the reach of any single agent.

An agent with these capabilities must not only be able to integrate reactive and deliberative reasoning when carrying out its activities; it must actively reason about the behaviour of other agents, be able to break down complex tasks and share the work involved with other agents, and must be able to integrate the results of tasks that are shared with other agents.

Evans and Anderson [1989, 1990; Evans et al., 1992] have developed a model of multi-agent planning dealing with these issues through the application of constraint-directed reasoning. Each agent is designed to function as one member of a group, and has a specific view of the group that varies with the agent's role and function. The agent itself relies on stored plans, and coordination knowledge is represented as sets of constraints and corresponding constraint relaxations that can be applied through negotiation when conflicts and inconsistencies arise. Coordination regimes defined in the model allow each agent to reason about local and global planning by selecting applicable knowledge and satisfying constraints; when conflicts arise agents cooperate to reformulate problem decompositions, task descriptions and distributions, and the integration of results.

Multi-agent planning within this architecture is accomplished through the exchange of *requests*: formal structures describing tasks. Upon accepting a request from some other agent, an agent employs a blackboard approach to

decompose the task into simple tasks that it can carry out itself or pass on to other agents. The agent applies knowledge in the form of independent knowledge sources to break the task down over time into a tree structure, whose root is the initial request, and each internal node of which represents a subtask decomposed from the original request. The leaves of this tree structure represent subtasks for which the agent has an existing plan or the knowledge that some other agent can handle the subtask. The control regimes employed by this architecture (controlling both the application of knowledge sources within an agent and the negotiation between agents) are based entirely in constraint-directed reasoning.

Aside from the general similarity in their basis in constraint-directed reasoning, there are two specific applications of constraints in the architecture of Evans et al. that have direct similarity to methods employed by the Waffler architecture. The first of these is the method used to control the application of knowledge sources during the planning process within an agent. At any point in time the agent may have many requests in various stages of decomposition, various components that can be distributed to other agents, and various results to be integrated. Knowledge sources directly support these processes, and so at any point in time there will be many knowledge sources available for execution, only one of which can be chosen.

In order to select the most appropriate knowledge source, each node in the plan tree is given a utility rating. Each knowledge source has an associated constraint on minimum utility (the utility the task must have before the knowledge source is considered to be applicable) and a constraint on minimum compatibility (a restriction of how compatible the task must be with the purpose of the knowledge source in order for it to be applicable). These constraints effectively limit the number of knowledge sources that are applicable at any given time, in order to limit the agent's deliberations. In the event that no applicable knowledge source can be found, these constraints can be relaxed.

Constraint relaxations are handled in a similar fashion. Some constraint in a request given to an agent may make that request incompatible with the agent's abilities or its current activities. In this case the agent can suggest relaxations to the constraint, which may or may not be acceptable to the requesting agent. Each relaxation has ratings of utility and effort, indicating the usefulness of the relaxation and the amount of effort associated with it, and constraints are employed in a manner similar to that above to control the applicability of constraint relaxations.

This is in effect the same methodology employed in the Waffler architecture to select the most appropriate action. The general concept of ranking alternatives against a scale and placing a limitation to exclude inappropriate alternatives is by no means novel. This similarity, however, is minor in comparison with the many advances employed by the Waffler architecture.

The most significant of these is that constraints in the Waffler architecture are viewed in a much more realistic manner than that used in [Evans and Anderson, 1989, 1990; Evans et al., 1992]. In the work of Evans and Anderson, constraints were used to calculate the utility ratings of each subtask available to an agent, and it was assumed that it was possible to handle the contributions of each and every constraint in the agent's knowledge base before making a decision. This is clearly impossible in everyday activities, and some means must be made to limit the applicability of constraints. This was dealt with in the Waffler architecture by creating a limited working memory and the ability to move relevant information in and out of working memory quickly.

The assumption that all constraints could be dealt with at once regardless of their number requires more than simply a similar working memory addition to the architecture of Evans and Anderson. Because all constraints could be processed at once, the agent was assumed to have completely accurate measures of utility. Again, as described Chapters 4 and 5, this is entirely unrealistic in the realm of everyday activities. The agent has at best an initial guess, and must consult its knowledge over time to extend the accuracy of that guess. This is dealt with in the Waffler architecture through the constraint-directed control regime that selects an appropriate action over time. If no action is appropriate, the agent will have further time to retrieve new concepts into working memory and thereby extend the accuracy of its measures, as well as possibly generating entirely new alternatives and taking in further information from the world around itself.

The architecture of Evans and Anderson also contains only the most rudimentary abilities to handle planning in dynamic domains. The agent posts expectations (using constraints) that are assumed to be confirmed or denied through perception. Aside from this, however, the architecture of Evans and Anderson simply assumes a plan can be easily carried out once it has begun. The plan either succeeds or fails, and can be retried or abandoned. As this dissertation has amply demonstrated, such an approach is impossible for use with everyday activities. Indeed, all of the work of improvisation involves adapting a general plan as time unfolds.

Constraints are also used much more extensively in the Waffler architecture than in the architecture of Evans and Anderson. While the latter employs extensive constraint-directed control regimes, and uses constraints to represent aspects of the agent's abilities and the tasks given them in order to make use of these control regimes, the Waffler architecture goes much further. Not only are constraints used to control the extent of improvisation, they are employed to represent the extensive regularities of the complex world around the agent. This encompasses the representation of intentions, as a well as the agent's extensive collection of background knowledge. The plans on which Evans and Anderson's agents rely, on the other hand, are static and assumed to be simple STRIPS [Fikes and Nilsson, 1971] plans.

Like all the other architectures and systems to which the Waffler architecture has been compared in this Chapter, none of these comparisons in any way invalidates the architecture of Evans and Anderson. The architecture of Evans and Anderson is clearly superior to the Waffler architecture in dealing with multi-agent interactions. Indeed, while the Waffler architecture allows the agent to adopt intentions and carry them out in the presence of other agents (in spite of any interference of those agents, and with the ability to take advantage of any positive influences provided by those agents), it contains no explicit means of facilitating cooperation between multiple agents. The Waffler architecture and that of Evans and Anderson are simply developed for completely different environments and purposes. The architecture of Evans and Anderson was never intended to cope with everyday activities, and while extensively addressing the needs of cooperating agents, has ignored many features (e.g. real-time reasoning) necessary to deal with everyday activities.

Far from being similar, the differences between these two architectures complement one another very well. Everyday activities, the bulk of human behaviour, require improvisation, and working in conjunction with other agents is a part of some everyday activities as well as many activities outside this realm. Although no research has yet been performed to this end, I contend that common social interactions can be considered everyday activities, and thus that at least some multi-agent everyday activities can be handled through appropriate representation as intentions in a Waffler agent. This represents an important avenue of future research, and shows promise in making extensive contributions to both improvisation and multi-agent planning.

8.6. Summary

This Chapter has examined several architectures and systems for two purposes: to illustrate the relationship between the components of the Waffler architecture presented in Chapter 5 and previous research, and to draw distinctions between these previous architectures and systems and the Waffler architecture. While much of this Chapter is critical of these previous architectures and systems, it must be stressed once again that any criticism of a particular system is strictly from the point of view of its application to everyday activities. Explaining cognition in everyday activities was not the intent of any of the architectures and systems described in this Chapter, and each performs extremely well within the bounds of the purposes for which it was intended.

In drawing distinctions between the Waffler architecture and previous systems, this Chapter has also has also served to emphasize some of the distinct contributions of the Waffler architecture. The next Chapter more extensively summarizes the contributions of the Waffler architecture and the associated research presented in this dissertation, and describes limitations and future research.

CHAPTER 9

DISCUSSION AND ANALYSIS

From the standpoint of computer science, AI has largely been the territory on the other side of an advancing frontier; research in AI extends the field of computer research, yielding to it domains once, but no longer, associated with AI.

– Dehn and Schank, *Artificial and Human Intelligence*

"Have we reached the end?" asked Piglet.
"Yes," I replied. "I suppose so."
"It *seems* to be the end," said Pooh.
"It does. And yet –"
"Yes, Piglet?"
"For me, it also seems like a beginning."

– Benjamin Hoff, *The Te of Piglet*

9.0. Introduction

The process of scientific research is based on incremental development and evaluation. Observed phenomena that are poorly explained by existing theories spur research, which leads to the development of new theories intended to provide a better explanation. These new theories are evaluated through comparison with the explanations provided by existing theories, and may then supplement or supplant previous theories. Each new theory is in a way the sum of all previous research, in that the questions driving the theory and the results obtained are dependent upon all previous research in the field. Each new theory also offers its own contributions, and becomes an addition to the base of theories that drive future research.

Previous Chapters of this dissertation have outlined the questions that arise from examining existing AI-based theories of activity from the perspective of everyday activities, and have presented a theory of improvisation, designed to explain phenomena that previous theories cannot. Previous Chapters have also attempted to show the connections between this research and that of others in the field, as well as the connections between this research and the generations of theories that lie behind it. This Chapter describes the connections between the work described in this research and the future: outlining the general and specific contributions of this work to artificial intelligence, and describing some of the questions that arise from this research that will drive future work.

9.1. Evaluating Research in Artificial Intelligence

The evaluation of scientific research has been characterised above as entirely dependent upon generally accepted theories to provide a metric for comparison. In practice, the concept of a theory for these purposes is broad. In cosmology, for example, existing theories of planetary motion have historically left discrepancies between predicted and observed astronomical phenomena. In engineering, existing devices have limitations in function or performance that provide comparison with the performance of new devices or technologies.

Evaluating research in artificial intelligence is not as simple as this. As described in Chapter 1, AI has no broadly accepted theories to provide a stable metric for comparison, as is the case in physics, chemistry, and other more mature sciences. Indeed, AI has far to go before such a general theory will be available: the problem of combining a significant portion of the myriad research areas that underlie intelligent agency into a larger theory alone will require solving most of the open problems that AI faces.

Having no general theory for comparison, AI researchers take many different approaches to evaluation. As AI is an interdisciplinary field, many of these approaches come from the disciplines that have made contributions in the past. Chapman [1990] points out that AI has divided itself into many schools, each of which has its own criteria for evaluation based on the field with which it aligns itself. There are those, for example, who argue that the only acceptable results in AI are those that are mathematically provable, and that research without mathematically provable theorems is beneath regard. There are those who argue that only completely embodied mechanistic

intelligences, regardless of their simplicity, can be viewed as concrete results. This research does not present results that satisfy either of these criteria.

Indeed, from the point of view of most of the many diverse fields that make up artificial intelligence, this research offers little in the way of concrete results: there are no theorems, no physical artifacts, no extensive experiments comparing improvisation to other methods of planning under strict controls, and no complete programs. Indeed, only the most basic of implementations has been provided. What this research does contain, however, transcends all off these. What has been described and demonstrated in this dissertation is an *approach:* a way of looking at a problem and a framework for dealing with that problem. Classical planning is one approach to activity: a framework for viewing the mechanics of intelligence as applied to activity, and a methodology for implementing those mechanics. Those working from a classical planning approach do not attempt to argue that it is a complete theory, only that it explains a set of phenomena, and can be expanded upon to answer further questions.[108] I argue that improvisation is an approach in the same sense as classical or universal planning. Improvisation is an explanation of how human intelligence is applied in the course of everyday activities, and the framework described in this dissertation is a methodology for implementing improvising agents. It does not answer all the questions of the entire spectrum of human activity, nor all the questions about how the cognitive methods employed during improvisation relate to those employed elsewhere. *It offers a framework from which to view activity, and from which to perform further research.*

Chapman [1990] argues that the majority of the most important works in AI to date are approaches, and characterizes the field as an approach-based science. He describes the place of the approach in artificial intelligence as follows[109]:

> AI has, thus far, produced no theorems important enough to be commonly known by name in the field, let alone in computer science and mathematics generally. With the exception of work in early vision, AI has produced few scientific theories that have withstood testing; and indeed few AI papers report experiments that would be recognized as rigourous by a biologist. AI has produced many technologies, but few have been demonstrated as superior to alternatives. It is common for

---

108 Indeed, many have done this: compare early classical planning work [Tate, 1976] with modern applications [Wilkins, 1988].

109 Emphasis mine.

> mainstream computer scientists to say of AI that the few worthwhile programs it has produced could have equally well been implemented in C using standard software engineering techniques and that the bulk of work in the field has no obvious application. *This is quite true, but misses the point: AI technologies are for the most part not interesting for their short-term applications but for their role in a broader view of what it takes to create intelligence. That is, their value derives from the approaches that gave rise to them.*

The importance of the approach to artificial intelligence relates strongly to its status as an immature science. While there is no general theory to serve as a firm basis for future research in AI, it is difficult to see how an individual theorem, program, artifact, or statistic works toward providing one. An approach, on the other hand, serves as a broad basis for experimentation, as a framework for examining all the other forms of results that AI research provides. Chapman [1990] cites Minsky's [1981] conceptualization of frames, and Hayes' [1985] work on naive physics as examples of approaches that have become seminal to the field.

The importance of approaches to AI has often been downplayed, and the significance of a new approach is often not appreciated until many years after its proposal. However, approaches are often the most lasting contribution of any research project in AI. Consider, for example, the DARPA speech initiative of 1971-1976 [Barr and Feigenbaum, 1981]. This initiative began with a strict set of guidelines by which systems were to be judged, and was expected to spawn many advances in speech understanding technology. Today, however, the initiative is not remembered for significant success in speech understanding, but rather by the approaches that it spawned: the blackboard approach to problem-solving, and early approaches advocating the extensive use of compiled knowledge that eventually gave rise to universal planning.

The fact that improvisation can be characterised as an approach, and that approaches in general are central to AI does not yet provide a basis for the evaluation of the research presented in this dissertation. As already mentioned, because AI is a multidisciplinary approach, it has borrowed the criteria of many of its founding fields to evaluate its work. I believe, as do others (e.g. [Simon, 1993]), that the formal methods of evaluation contributed by other fields help AI only to the extent that the research involved relies on that field, and in many cases may actually stifle progress in AI. Consider for example, the Tileworld (Section 6.3.2) experiments of Kinney and Georgeff [1991], which examine the bold vs. cautious option filtration strategies in IRMA, described in Section 8.2. These experiments attempted to illustrate how the degree of boldness of an agent (its willingness to consider new

alternatives) affected its performance in the Tileworld environment under varying degrees of dynamic change. Four reaction strategies were used: blind commitment to filling a particular hole; changing goals when the target hole disappears; changing goals when the target hole disappears or a new hole is found; and changing goals when the target disappears or a nearer hole is found.

This experiment illustrated that for most rates of change, being open to new options when either a nearer hole appears or the target disappears results in the most effective agent. But how does this translate to a more realistic domain, such as the kitchen domain used in the experiments in Chapter 7? Does this result have value to more complex domains, where many more strategies are available, or where such a precise division of reaction strategies is precluded? Quantitative results such as this give the appearance of concrete answers to hard problems, but in reality, the concrete answers are the result of simplifying and controlling the problem to the point where the results are inapplicable to any but the original simplistic domain[110].

Evaluating AI research solely by virtue of specific results such as that described above also raises larger questions: What does this result bring to AI? How does it help AI toward the general theory it seeks? While by no means all of the quantitative results being obtained in AI today are as ambiguous as that above, and while quantitative results certainly have their place in AI just as any other method of evaluation, there is a grave danger in the current trend in AI to make such results the central feature in evaluating a course of research. Demanding specific, quantified, formal results from *all* research in AI is too presumptive, I argue, for the current state of the field. While evaluation based on precise, quantified results is valid in some cases, too often the narrowness of the result is not emphasized or even taken into account (such as in the case described above). Were these quantified results resting upon the foundation of a larger, generally accepted theory of how intelligence works, they would be of great benefit. However, AI is not yet at this point. Most research dealing with approach-based questions, of direct and present concern to AI cannot hope to return specific, quantified results, at

---

110 This is by no means an isolated case. It is not difficult to construct controlled experimental domains in which agent designs can be compared. However, in many cases, very slight changes in the domain or the internal biases of the agent can so greatly affect these results, and the failure to acknowledge this fact makes these results almost meaningless [Anderson and Evans, 1994].

least at this stage, without simplifying the questions that drive such research to the point of being non-issues.

Despite the trend toward formalization and quantitative results, there are still those who recognize that the ill-defined nature of problems in AI and the broad spectrum of research in AI require more qualitative means of evaluation. In evaluating distributed problem-solving research for example, Decker et al. [1989] emphasize that quantitative comparison of systems and approaches is unworkable, both because of the difference in emphasis and focus between research projects, and because of the qualitative nature of many of the dimensions that are used to draw comparison. Instead, Decker et al. provide a list of questions that can be used to provide a basis for evaluating how well a system addresses the important issues of the field. This has its own limitations, in that only broad comparisons can be drawn, due to the ill-defined nature of the domain and the broad spectrum of research projects. However, it is precisely the type of evaluation that AI needs, given the current state-of-the-art. Given that AI has no general theory to provide a metric for comparison, the field must fall back on the principles of scientific endeavour that underlie research in more mature fields, and upon which those fields too depended more directly in their immaturity. When confronted with a new theory, instead of expecting provably correct results or completely implemented mechanics, we must question the basic theory itself:

- Does the theory explain the phenomena it purports to?
- Are these phenomena significant?
- Does the theory attempt attempt to demonstrate or replicate these phenomena under its guidelines?
- Does the theory emphasise its weaknesses as well as its strengths?

Like all science, developing a comprehensive theory of intelligence must in the end rely on these basic evaluation methods.

The remainder of this Chapter describes the contributions of this dissertation in light of questions such as these, and describes future work.

9.2. Contributions

This dissertation began with the argument that existing theories of activity, while being well-suited to the specialized types of human activity for which

they were originally designed, were inadequate for dealing with everyday activities. An analysis of everyday activities illustrated that these activities are *complex activities whose common and mundane appearance is a result of the extensive experience of the agent performing them.* Through the analysis of specific episodes of everyday activities, Chapter 2 illustrated that everyday activities are in fact much more complex than universal planning proponents have argued in recent years. Chapter 2 also presented a set of characteristics common to everyday activities, characteristics that must be accounted for by any agent architecture designed to deal with this type of activity. Beyond stating general characteristics, this Chapter has also shown the connection between everyday activities and other types of human activities (e.g. expert activities, pure planning, pure reactivity), and has illustrated the placement of everyday activities on spectrum of human behaviour.

Chapter 3 has examined existing planning approaches in terms of these characteristics. Chapter 3 has also argued that one of the major current directions of AI planning research, extending classical and universal approaches to deal with complex everyday activities, is not likely to prove fruitful for the simple reason that these approaches were designed for types of activities far removed from everyday activities. In particular, Chapter 3 has argued that both the classical and universal planning approaches are too inflexible, overconstraining, and too heavily reliant on static compiled knowledge to be useful in everyday activities.

Having established the need for a new approach to dealing with everyday activities, Chapter 4 has presented *improvisation,* a new approach designed specifically for the characteristics of everyday activities described in Chapter 2. Improvisation relies on using plans as guides to activity rather than dictators of it. An improvising agent possesses both routines for everyday activities, representing knowledge that has been applied successfully many times in the past, as well as the diverse background knowledge from which the agent's routines have been compiled. The ability to access this background knowledge allows an improvising agent to apply its routine flexibly, to cope with differences in setting and dynamic rearrangement of the outside world that would confound previous approaches. Chapter 4 has illustrated that the improvisational approach explains the phenomena described in Chapter 2 to a far better degree than any approach to date. This Chapter has also argued that constraints are an ideal mechanism for representing both the routine and background knowledge that the agent applies to guide it through its course of activities.

Chapter 5 has described the embodiment of this improvisational approach in the form of an architecture for improvising agents known as the *Waffler* architecture. This Chapter has extensively discussed the nature of constraints in everyday activities, and has described precisely the types of constraints required to implement the necessary routine and background knowledge necessary for improvisation in everyday activities. Chapter 5 has described the knowledge structuring mechanisms required for improvisation, and in particular has presented *intentions:* knowledge structures that serve both to maintain the routine knowledge that forms the backbone of improvisation as well as the background knowledge necessary to apply the agent's routine in varying situations.

Chapter 5 has also described the processing of these knowledge structures within the Waffler architecture, through its major components: perceptual and effectory components; a limited working memory consisting of the agent's intentions and conceptual knowledge relevant to the current situation in which the agent finds itself; a long-term memory consisting of the agent's entire store of knowledge of activity and the world around itself; and a deliberation component that allows the agent to repeatedly select the best action available to it. Because of the extent of the agent's knowledge and the general and ubiquitous nature of improvisation, much of the processing described in Chapter 5 is devoted to limiting the agent's consideration of knowledge to that most relevant to its current activities. Chapter 5 describes the memory migration process and the constraint-directed control regimes that make this possible.

Creating an implementation of this architecture required the adoption of a simulation testbed that would both provide an artificial world for the agent to inhabit and a platform with which to iteratively prototype the computational processes making up the agent itself. Chapter 6 has described the analysis of existing intelligent agent simulation systems and the reasons why each of these was found to be inadequate for these purposes. Chapter 6 describes in detail the development and structure of *Gensim,* a new simulation system designed to support the iterative development of intelligent agents and simulated environments within which these agents can be examined, tested, and evaluated. Gensim is generic and modular, and is applicable to a wider range of agent designs and environments than previous simulation systems.

Chapter 7 has described a partial implementation of a Waffler agent using the Gensim simulator. This Chapter has described a particular environment in which the improvised tea-making behaviour described throughout this dissertation can be displayed and experimented with. The use of the facilities

provided by Gensim to simulate the extensive parallelism displayed in the Waffler architecture was also described, as was the implemented agent engaged in three complex instances of improvised behaviour. These instances by no means illustrate the complete range of abilities of the Waffler architecture, but were selected for their ability to demonstrate the major characteristics outlined in Chapter 2. The actual output of the simulator illustrating the agent's behaviour in each of these examples is contained in the Appendices following this Chapter.

Each of these Chapters contains important contributions to the field of artificial intelligence. Although some limited analysis of everyday activities has been performed in the past, the analysis and resulting characteristics presented in Chapter 2 is unique in its comprehensiveness. Similarly, while the universal and classical planning approaches have been critiqued since their inception, the comprehensive analysis of these approaches in terms of the characteristics of everyday activities is also unique.

Despite the use of the term previously in the AI literature, I have already illustrated in Chapter 4 that the approach I have termed improvisation and its embodiment in the Waffler architecture is unique *in that has a concrete basis for improvisation*, by focusing both on the agent's compiled routines and the background knowledge from which those routines have arisen. In addition to this new approach to activity, this dissertation has contributed two software systems: the Gensim simulation system and the implemented Waffler agents used in the examples described in Chapter 7. While the approach behind these implemented examples is the most significant contribution of this dissertation, the examples themselves serve as physical evidence for the ability of the Waffler architecture to demonstrate the phenomena of everyday activities described in Chapter 2. This in turn validates the claim that the improvised approach is more suitable to everyday activities than the approaches critiqued in Chapter 3.

The Waffler architecture and the improvisational approach behind it form the main focus of this dissertation. However, the research leading to the Gensim system (Chapter 6) both directly supports this focus and forms a separate line of research on its own. The direct motivation behind the development of Gensim was the need for a platform upon which to begin implementation of a Waffler agent. The realization that such a flexible platform would also be needed elsewhere, however, led to the decision to make Gensim as generic as possible in order to support a wide range of agent designs. This in turn led to many of the unique contributions of Gensim itself, apart from its direct support of the implementation of the Waffler

architecture. The major contributions of both of the Waffler architecture and Gensim are described in the following Sections.

9.2.1. Contributions of the Improvisational approach

In a recent review of modern reactive planning research, Lyons and Hendriks [1992] describe the following open areas as major goals of future research:

- There has been little work in integrating reactive work with the real-time computation field.

- Resources play a key role in many reactive domains - the concept of 'making do' with what is available - but as yet there is no unique reactive viewpoint on resources.

- There are a number of suggestions, but no definitive answers yet, on an appropriate way to view long-term 'planning' so as to integrate it with reaction.

The main contribution of the improvisational approach and its embodiment in the Waffler architecture is in addressing these issues. Improvisation illustrates how a plan may be used as a resource, together with the background knowledge from which that plan was derived, to provide robust, flexible behaviour in complex but familiar environments. Improvisation is also the epitome of "making do": a Waffler agent makes the best decisions possible given the routine and background knowledge available to it and the amount of time available to explore that knowledge. That is, a Waffler agent performs in a satisficing manner given a situation requiring variations on its normal routine, as well as in situations where time is limited. Improvisation also seamlessly integrates responses recommended by plans and reactions supplied by background knowledge. Moreover, the knowledge used in improvisation is *explicit*. Rather than burying the agent's knowledge of activity and the world around itself in a pre-compiled network, an improvising agent can be observed in its reasoning, and the explicit routine and background knowledge can be restructured and applied in new situations. This will more easily support experimentation in combining improvisation with other intellectual abilities (e.g. learning).

In comparison to the models critiqued in Chapter 3, improvisation provides all the responsiveness of a universal planner, but without the assumption of completeness. Rather than making immediate decisions and assuming a

completely compiled arbitration (response) function, an improvising agent makes the best decisions possible given the knowledge and time at its disposal. The best decision given a particular situation may not occur to the agent immediately: the situation may not be part of the agent's routine and may thus require exploration of background knowledge to find an appropriate response. In such a situation, a universal planner containing the situation as part of its universal plan would indeed perform better than an improvising agent. However, as argued in Chapter 3, there is no way to completely compile a universal plan for the complex domains and the amount of variation in everyday activities, despite their routine appearance. Improvisation is thus the more viable and practical approach in everyday activities.

When compared to the classical planning model, improvisation is by far more responsive and flexible. As described in Chapter 3, classical planning has no responsive element naturally associated with it, and was designed for a completely different kind of activity. Improvisation provides for both the use of plans and the integration of those plans with background knowledge, striking a balance between the use of compiled knowledge in the form of plans and the consideration of the alternatives that arise dynamically from the world around the agent.

The comparison to universal and classical planning serves as a reminder of an important issue in the evaluation of improvisation that deserves mention even before any specific weaknesses are dealt with. Just as classical and universal planning were originally designed for specific parts of the spectrum of human activity, so is improvisation. Improvisation is not suited to detailed planning tasks, to which classical planners are well-suited for example. Improvisation is designed specifically for everyday activities, and is best suited to those particular activities. If no part of some activity is ever routine to an agent, and the agent must rely strictly on its background knowledge, an approach relying on strict planning will be more appropriate. However, I have argued in Chapter 2 that everyday activities occupy a wider portion of the spectrum of activity than those to which classical and universal planning are suited. In Chapter 4, I have also illustrated the connections between everyday activities and other forms of activity (e.g. that pure reactivity is subsumed by improvisation, and that higher forms of improvisation require greater and greater reliance on background knowledge that eventually becomes indistinguishable from planning). Moreover, improvisation is a satisficing approach, and simply performs less optimally as more and more reliance on background knowledge is required. Both classical and universal planning, on the other hand, are rigid, and break down rapidly

as they become further and further removed from the types of activities for which they were originally intended.

This flexibility is one of the most important contributions of the improvised approach. Almost since its inception, AI has been known for few characteristics more so than its brittleness [Dreyfus, 1981; Lenat et al., 1986]. Human performance at tasks requiring intelligence can be characterized as flexible and satisficing: each of us has a collection of tasks about which we know a great deal and at which we can perform expertly, as well as tasks about which we know less and less and which we perform correspondingly more poorly. Intelligent systems on the other hand, and especially those performing symbolic reasoning, are well known for hard-and-fast boundaries before which they know a great deal and after which they know virtually nothing.

Improvisation, however, emphasises satisficing, as I have already stressed above. Improvisation makes no claims of optimal performance, and instead does as well as it can with the resources available to it. While once again, the Waffler architecture is designed for everyday activities, and is by no means an architecture for general intelligence even approaching the human level, *it does demonstrate the kind of flexibility that is exemplified in human information processing.* More than any previous approach to everyday activity, the improvisational approach has the potential to make great inroads toward an architecture for more general intelligence.

The Waffler architecture also makes contributions to artificial intelligence by emphasising rationality beyond the strictly decision-theoretic. While employing quantitative utility factors in its reasoning in the implementation presented in Chapter 7, these are secondary in the Waffler architecture[111]. The primary force behind rational behaviour in a Waffler agent is not quantitatively reasoning about utilities, but the triggering of the recall of concepts and the appropriate activation of constraints. The bulk of an improvising agent's rationality is its basis in experience: routines make sense to follow because they've worked many times in the past. AI is only now just beginning to recognize that not all of human reasoning, and especially reasoning in everyday activities, fits nicely into the decision-theoretic model

---

111 Indeed, the utility factors used in the examples described in Chapter 7 were purposely kept simple primarily to illustrate how far an agent can go with only simple quantitative measures of utility. These examples could just as effectively be implemented using qualitative measures of utility.

of rationality (Section 2.4). Even decision-theorists have cautioned over the years on over-reliance on quantitative reasoning and have encouraged more reliance on the experience, intuition, and qualitative judgements that characterize human reasoning[112]. As Simon [1969] states:

> Numbers are not the name of this game, but rather representational structures that permit functional reasoning, however qualitative it may be.

I argue that previous experience, intuition, and qualitative judgements characteristic of improvisation are the some of most important aspects of human decision-making, and must occupy a central place in artificial intelligence if real progress in the field is to be made.

The improvisational approach that forms the main focus of this dissertation also makes major contributions to constraint-directed reasoning. The Waffler architecture has emphasised the use of constraints to directly drive activity, rather than focusing on the violation of constraints as most previous research has done. It has demonstrated that constraints have the power and flexibility to provide the bulk of the extensive knowledge representation mechanisms required by improvisation. In this fashion, this work can be viewed as an extension of Fox's [1983] work on constraint-directed reasoning in scheduling: extending the understanding, the variety of application, and the power and utility of constraint-directed reasoning.

The connection between constraint-directed reasoning makes an additional significant contribution to AI. In earlier work on constraint-directed reasoning, Fox and Nadel [1989] argue that constraint-guided search and heuristic search can form a unified theory of problem-solving. This dissertation, in demonstrating the strong connection between improvisation and heuristic search and the even more intimate connection between constraint-directed reasoning and improvisation, helps to support this argument and offers a strong potential for promising future research toward this end.

Finally, the improvisational approach and the Waffler architecture contribute by emphasizing the view that *much of intelligence involves taking advantage of the structure of the world around the agent.* During the course of time over which this research has taken place, others in the field have begun to see this emphasis as well and are using this view to develop new

---

112 See [Ackoff, 1981; Hammond, 1974; Mintzberg et al., 1976; Ricketts, 1979]

approaches to other problems. For example, Miller [1994] has recently argued that perception is in many cases not the hard problem it has been made out to be, simply because humans in many cases take advantage of "secondary sensing", the ability to convey complex ideas using simple sensing by taking advantage of structure in the world around the agent:

> When a human is in a natural setting (behind the wheel of a Honda Accord, going 20 miles over the legal limit on a 2-lane road...), neither nature nor the Federal Highway Commission (FHC) would expect him or her to be able to interpret the complex images and kinematic sensory output that would be needed to determine the layout of the road ahead. FHC has had a trained professional go down every road as it is laid at the speed for which it was laid and convert this huge amount of complex sensory data into a small set of pithy, easily parsable road signs that are all that we can be expected to deal with during high-speed travel. The driver does not have to be able to determine the rate of turn in the road or separate road from shoulder. FHC has already done it. FHC then masticates the sensory data into small, easily swallowed pills that can easily be digested by the computational capabilities of the most average of drivers. Bright-yellow reflective lines separate road from shoulder. Iconic and text representations tell the driver that the road turns left or right. The driver doesn't need to be able to see if the cross-traffic has a stop sign; he or she knows: he or she has been given the predigested sensing of the FHC technician in the form of an easily machine-parsable sign that says that the cross-traffic must stop...This is sensing the way nature intended it - no muss, no fuss, no strain on the brain.

This agrees profoundly with everything I have attempted to argue in this dissertation, and one of the most important contributions of this work is in further emphasizing the concept of intelligence as taking advantage of structure in the world and acting as a broad approach for further experimentation along these lines.

The most obvious value of the improvisational approach and the Waffler architecture is in a demonstration of how an intelligent agent can cope with the demanding nature of everyday activities. As I argued in the outset of this dissertation, the ability to deal with everyday activities has long been one of the major goals of artificial intelligence. In moving toward this goal, this research has already made significant contributions to the field.

However, there is also great potential practical value beyond these academic contributions. Improvisation involves the flexible application of routines during the course of activities which are at the same time both familiar and variable. This scenario occurs widely in the real world. In manufacturing, for example, those tasks which require rigourously specific repetitive actions have been extensively automated in today's world. The technology presented here would allow automation to move toward activities that are routine but also contain variations in individual activity that must be dealt with

dynamically and may be unpredictable. Indeed, it provides a method of applying routine knowledge flexibly anywhere that a satisficing approach can be tolerated. I would argue that since humans as satisficing intelligent entities already do the bulk of this work, a satisficing approach is tolerated in these tasks to a much greater degree than is likely to be admitted.

Being a model for dealing with everyday activities, the Waffler architecture also has applicability anywhere a model of human reasoning is needed. This model could be used, for example, to allow a computer system to interact with its users in a more human manner. The Waffler architecture can also be used to model human reasoning in everyday situations, allowing the inclusion of humans in complex simulation models for research in a wide variety of areas, including natural resource management [Anderson and Evans, 1994].

Finally, in terms of modelling human reasoning in everyday activities, this approach has the potential to have the same kind of impact that the modelling of expert reasoning has had on the field. The ability to computerize the understanding of everyday activities has the same ultimate benefit as that of expert reasoning: making computer technology more useful to mankind by extending the range of problems to which computer technology can be applied[113]. The more concrete benefits are also similar to those of modelling expert reasoning: for example, freeing people from mundane activities and expanding, preserving, and circulating experience-based knowledge of a particular activity. A more thorough understanding of cognition in everyday activities will result in a new generation of intelligent systems with the ability to deal with the kind of mundane reasoning that we take for granted daily, but which has eluded AI since its inception.

As mentioned at the beginning of Section 9.2, while improvisation and the Waffler architecture form the primary focus of this dissertation, many additional contributions are made by the Gensim simulation system developed to support this research. These contributions are outlined in the following Section.

9.2.2. Contributions of Gensim

The most obvious contribution of Gensim is the software itself, a simulation testbed that allows the development, evaluation, and testing of a wider range

---

[113] Winston [1984] argues that this is the primary goal of AI.

of agent designs and simulated environments than that supported by any intelligent agent simulation system to date. The particular features of this system which make it unique and can be considered contributions to the field are:

- Gensim's modularity and overall generic nature, allowing the independent definition and use of agents and simulated domains;

- The system's object-level interface, modelling perception at a high level and striking a balance between sophistication and ease-of-use in order to make the system applicable to a wide range of projects;

- Gensim's emphasis on the separation of agent and simulator knowledge, forcing the agent to rely on its own knowledge and perceptual abilities and thereby creating a more realistic simulation;

- Clear and explicit definitions of actions, events, and agent timing, allowing users to quickly gauge the fit of their work to Gensim's simulation abilities, and

- The ability to handle multiple agents within this framework, as well as agents containing multiple timeshared processes.

Chapter 6 has amply demonstrated the need for such a system, and the statement of requirements, critique of existing testbeds, and the arguments presented for using simulation to evaluate intelligent systems can themselves be considered significant contributions. Over and above the general contributions of Gensim as a software system, the characteristics upon which the system were based bring about some unique contributions on their own.

The foremost of these is the generic, modular nature of the system. Constructing modular domains in which various types of agents can be placed for examination and evaluation not only facilitates comparison between agent designs, it also allows those domains to be useful in situations other than the particular agent evaluation for which they were intended. That is, the simulated domain can be treated as a body of knowledge as opposed to a tool to be discarded after use, as Davis and Blumenthal [1991] argue should be the ideal place of models in scientific endeavour. By treating a model as a body of knowledge, models can be shared, preserved and modified, so that one simulated environment can contribute to the design of others.

Gensim also has significance to the field of artificial intelligence in a much broader sense: improved environments for examining intelligent agents will facilitate better and more direct comparison of intelligent agent designs. In a recent survey and analysis of simulation in AI [Hanks et al., 1993], Cohen describes three phases of research in which simulation plays a role. In an *exploratory* phase, a simulation testbed provides an environment in which agents can behave in interesting ways. In a *confirmatory* phase, the characterizations of behaviours of agents can be tightened through controlled experimentation employing a testbed, performing experiments designed to test specific hypotheses. In the third phase, *generalization*, the researcher attempts to replicate results, demonstrating the findings of confirmatory research in less confined settings. Cohen notes that complex simulators and agents are to be preferred for the first two phases, in that they can support more interesting forms of behaviour. However, of all the systems examined in this paper (as well as others described by [Hanks et al., 1993]), Gensim is the only system that can adequately support the generalization phase. There is much argument as to how generalization can be performed (for example, how some finding discovered through a given agent inhabiting the Tileworld can be shown to be a general feature of some type of activity). However, the most practical method is obvious: the same behaviour can be shown to be true in domains that differ widely across many features. In order to perform such generalization, simulation technology *must* support the development of widely-differing, modular domains, that can share a common interface between agents. Systems such as Gensim are a large step in this direction.

In addition to supporting a wide variety of domains and agents, Gensim also provides support for representing varying degrees of autonomy in agent architectures. Research in multi-agent planning systems has shown that agents in heterogeneous environments are rarely completely autonomous: the depend directly and indirectly on other agents (often those in superior social positions) in the environment they inhabit, and they similarly control and influence other agents. It is therefore important that an agent in a multi-agent system possess *flexible autonomy* [Evans and Anderson, 1990; Evans et al., 1992]: the ability to both operate fully autonomously and to depend on other agents to varying degrees. Gensim's facilities allow variations in autonomy along three lines: knowledge representation, concept sharing, and communication [Anderson and Evans, 1993]. Gensim's knowledge definition facilities allow agents to directly refer to and reason about the exact objects maintained by the simulator for objective purposes (severely limiting autonomy); allow the agents to share their own representations of domain objects between one another (moderate autonomy); and also support

complete independent world models for each agent (high autonomy). Domains can also be organized so that agents need only have a few concepts in common, such as a common reference for the location of objects (high autonomy), or must have common definitions of a large number of concepts (limiting autonomy). Finally, several communication methods for identifying objects referred to in agents' actions and sensory requests are available. Agents can simply specify the specific symbol or name of an object used by the simulator when referring to an object in an action or sensory request (low autonomy); can refer to its own name or symbol for the object, (moderate autonomy); or can specify objects in an indexical-functional or deictic [Agre, 1988] fashion, allowing an object to be described by its relationship to the agent itself or to other objects the agent knows about (high autonomy).

9.3. Limitations

While the improvisational approach, the Waffler architecture, and the Gensim simulation system all make significant contributions to the field of artificial intelligence, each also has its limitations. Many of these limitations are due to the nature of research in intelligent agency as described in Chapter 1: it is impossible to construct an intelligent agent without any assumptions about the myriad of intellectual faculties involved without solving each and every open problem facing artificial intelligence. Given the current state of knowledge in artificial intelligence, many assumptions must be made about the nature of cognitive components within an intelligent agent simply because a complete understanding of those components is not available.

Once again, because of their difference in focus, the limitations of Improvisation and the Waffler architecture are separated from those of Gensim, and are dealt with in the following Sections.

9.3.1. Limitations of the Improvisational Approach

As already mentioned, the most significant limitation of the improvised approach is its reliance on satisficing and the need for a routine. Improvisation cannot cope any better than classical planning with completely novel activities, nor is it appropriate for those activities where absolute optimality is required. However, as has been argued extensively in Chapter 2, the bulk of human behaviour is not burdened with either of these requirements. Improvisation does not require that an activity be absolutely routine, as universal planning does, nor that response time be a non-issue, as

classical planning does. Put simply, like other approaches before it, it is designed for a specific area on the spectrum of human activity described in Chapter 2, is extremely well-suited to that portion, and becomes less appropriate as activity moves into areas for which it was not designed.

This is not really a limitation *per se*, as a focus on everyday activities was the intended in the approach from the beginning. However, in terms of the Waffler architecture embodying the improvisational approach, many questions remain, despite the extent to which the architecture has been developed and implemented.

For example, while a Waffler agent relies extensively on its working and long-term memories, some implementational questions in regard to these have been left unanswered. Chapter 5 has focused on describing the requirements and operation of working memory and long term memory from the perspective of improvisation in general, rather than from the point of view of any specific instance of improvised behaviour. Thus questions about the length of recall and whether a newly-perceived object should be put into long-term memory are omitted. I avoided questions such as these primarily because they cloud the more important issue of how improvised behaviour works, but also because contradictory examples concerning these issues can often be found. For example, we may see some object in the context of our activities once and remember it forever, another time it may be forgotten immediately or partially remembered.

A more significant issue that remains open is the nature of knowledge representation in the extensive and wide-ranging background knowledge available to the agent. Once again, I have avoided attempting to deal with every possible knowledge representation problem in order to focus on those aspects of knowledge representation that are unique or particular to improvisation. I admit that there are staggering knowledge representation difficulties that must be overcome in order to support the kind of wide-ranging background knowledge required by improvisation in everyday activities. However, work on large-scale knowledge representation projects such as CYC [Lenat et al., 1986; Guha and Lenat, 1990] has already made major advances in the state of knowledge representation mechanisms in AI. Future work in this area will no doubt find new and better ways of dealing with these problems, greatly widening the scope of problems to which improvisation can be practically applied.

I have also omitted from improvisation and the Waffler architecture any component claiming to account for the origins of the agent's motivations.

While basic motivations must of course exist in any intelligent agent, I believe that the nature of these motivations go beyond the focus of improvisation, and that the motivations of an intelligent agent are much more complex than those in AI have previous assumed. To some degree, I believe that constraints can be used to model motivations in much the same manner as expectation constraints are employed in the Waffler architecture. One possible motivator defined for an agent, for example, could be *thirst* . If thirst is modelled as a real-valued measure ranging between, say, 1 and 10, a *motivational* constraint can then be defined on this measure requiring it to be less than a given limit. The violation of this constraint would then motivate activity to resolve the violation.

Such motivations are easily modelled, and it is possible to conceive of hunger and many other basic motivations in this manner. Indeed, this has been used in the construction of intelligent agents in simple environments, including improvising agents. Anderson and Evans [1994], for example, illustrate various types of intelligent agents in an wilderness setting that are motivated through hunger and fear in precisely this fashion. These motivational constraints can be thought of as a more sophisticated version of the policies used to motivate activity in Runner [Hammond et al., 1990][114]. The difficulty with employing this approach in complex domains for the purposes of improvisation is that many motivations are not nearly so clear as those above. The desire for tea, for example, arises not only from thirst but from a complex mixture of many factors: previous pleasurable experiences associated having tea, caffeine addiction, the desire for something hot, habit, and social custom, for example, can all come into play in addition to thirst. Many of these factors do not seem well-suited to the structure of motivational constraints defined above (e.g. it is difficult to conceive of a *desire-for-something-hot* motivator using the above framework). An alternative would be a *desire-for-tea* motivator, but this would lead to incredible numbers of such motivational constraints (one for every possible desire) and no easy way of maintaining a measurable level for these motivators. What, for example, makes the level of desiring something hot go up or down? This is not nearly as clear as it is for more basic motivations such as hunger or thirst.

While there are those that view even high-level motivations to follow this type of scheme (e.g. [Minsky, 1986]), it seems practically unworkable for the

---

114 See Sections 5.1.2. and 8.4.

purposes of everyday activities. What is really required to move from basic motivations such as hunger and thirst to more high-level complex combinations of motivations is is a more complex theory of the basis of motivation in emotion and social knowledge, and the inclusion of a model based on this theory into the improvisational approach. While not designed explicitly for motivations, Reilly and Bates [1992] have developed a computational model of emotions in an intelligent agent based on cognitive modelling work of [Ortony et al. 1988]. The model of Reilly and Bates is designed to supply emotional response in "believable agents": agents designed to display believable characteristics and personalities in their interactions with humans and with one another. While this model is not sufficient to supply the sophisticated kinds of motivations necessary for everyday activity, the emotions created by the model are used in a motivational sense in that they affect the decisions of the agent architecture of which they are a component. The application and extension of a model such as this to provide for motivations in an improvisational architecture is an important area of future research.

Another way in which the Waffler architecture is limited is in its exclusion of episodic memory. In Section 3.4, an extensive argument was made against the reliance on episodic memory in everyday activities, based on the reliance in everyday activities on the paradigmatic case of an activity as opposed to individual episodes of behaviour. However, we must also recognize that in everyday activities we *do* have the ability to recall components of previous episodes of behaviour and to make use of these in current activities. I have never claimed that episodic memory has no place in the performance of everyday activities; only that it should not be paramount over semantic memory. While there are those that argue that semantic memory can function completely without episodic memory [Tulving, 1972], both are important in human cognition. Indeed, there is a great deal of overlap between episodic and semantic memory, and in experimental situations it is impossible to completely isolate one from the other [Ashcraft, 1989]. Addressing the issue of including and relying on both semantic and episodic memory relies on a better understanding of both of these components than exists today, and would be an extremely ambitious research effort with a focus far beyond everyday activities.

The need for episodic memory is associated with another limitation of the approach presented in this dissertation: a lack of any model for the learning of routines. The ability to learn a routine is intimately connected with episodic memory: as episodes of behaviour are experienced, we gradually generalize aspects of that behaviour into routines. This is only a limitation of

the improvisational approach in a very broad sense. the lack of a model of learning is much the same from the perspective of improvisation as a lack of a complete model for perception: the inclusion of such a model would certainly aid in validating the improvisational approach; however it is a peripheral issue largely independent of improvisation and outside the scope of this dissertation.

In terms of the Waffler architecture itself, the major limitation undoubtedly lies in the technology assumed by the agent's working memory. In order to support improvisation, the agent's working memory must be able to immediately realize connections between concepts contained in the memory itself, and also must be able to activate any constraints contained in that memory in parallel. Now, this is by no means difficult from a conceptual point of view. The working memory in the Waffler architecture is assumed to be limited and small, and human working memory certainly has these parallel aspects. However, implementing such a memory is by no means simple. Once again, as the nature of human memory becomes better understood and parallel computing applications more widely prevalent in AI, the implementation of such a working memory will not be such a daunting task.

The implementation described in Chapter 7 uses the time-sharing abilities of Gensim in order to simulate this parallel working memory, to demonstrate its conceptual abilities. This implementation is also limited in many ways, mainly due to its prototypical nature already described in Chapter 7. For example, only small portions of the intentions used are implemented, only one action is allowed on any one cycle, and physical and mental actions cannot proceed in parallel. None of these limitations are technical however: they are simply choices that were made in the division of the limited resources available to this project.

9.3.2. Limitations of Gensim

The Gensim system also has its limitations, some of which are a result of the goals of the system itself and others which are a result of choices made during the design and implementation of the system.

One very legitimate criticism of Gensim is that it takes a great deal of time to define a domain for agents to inhabit. This is a direct result of the generic nature of the system. As argued in Section 6.2, there is an inverse relationship in computer science between flexibility and complexity. A system designed for a wide range of applications will be more complex and

require more work on the part of the user, since fewer assumptions about the use of the system can be made when implementing it.

Gensim is an extremely flexible simulator; much more so than any of the simulation systems reviewed in Chapter 6. The price of all this flexibility, however, is the requirement that the user define all agents, operations, and interactions supported by the domain. In many cases, this may be a very complex effort: defining all the interactions in a domain may take a great deal of time, simply because all the subtleties in the interaction of intelligent agents with a complex environment may not be immediately recognizable. Intricacies will often arise in a partial implementation of a domain, necessitating modifications from slight changes in the behaviour of objects to large-scale domain redevelopment. This limitation is unavoidable in principle in a simulator whose goal is to be generic. If the requirements of the agent and domain involved meet those of one of the more special-purpose simulators described in Chapter 6, it makes good sense to use that simulator. However, as I have argued in Chapter 6, very often a domain or agent design that may at first seem tailor-made to an existing simulation tool may only be found to be unsuitable after a partial implementation.

This dilemma is also a common one in expert systems development. It makes sense to use an existing expert system shell if the characteristics of the domain and task meet those of some existing system, as opposed to constructing the system from scratch. However, there is a strong element of risk involved in that domains are generally not well-understood at the outset, and the changing characteristics of the problem often invalidate the choice of tool. In these cases a more generic tool, requiring more work on the part of the user but providing the flexibility to support a changing domain or task is by far the better choice. As I have argued in Chapter 6, much research in intelligent agency falls into the latter category and provides a strong need for a system such as Gensim.

The most significant true limitation of Gensim is that it is timeshared and contains no true parallelism. While a small agent interval makes this timesharing as transparent as possible, there are still areas of research (e.g. fine-grained parallel actions) for which timesharing cannot be considered a suitable substitute. Even when such specialized areas are not considered, timesharing still presents some difficulties. As described in Section 6.4.1, the timesharing used in Gensim is organized to effectively interrupt the agent every P time units to give it new sensory information. However, this timesharing is done within Gensim itself, and an agent process can be interrupted at any point in time. This forces the user to design agent

processes with this in mind[115]. While true parallelism would be even more ideal, timesharing should be made an operating system function (i.e. agent processes implemented as independent OS processes rather than LISP functions) in order to make timesharing more transparent to the agents involved in the simulation.

Another limitation of Gensim is that agents are processed in a strict order without regard to their abilities: if three agents perform a particular action, the effects of one agent's action are adopted before the others simply because of the order in which the agents are defined to the simulator. This is limiting in multi-agent situations involving competition, where one agent's actions being considered before those of another can be very significant. The simulator would require a model of the agent's abilities in order to handle this in a more realistic manner.

Gensim also requires more support facilities for mental actions, such as adopting intentions. In the agent timing used by Gensim, each agent is given the sensory information previously requested, has its processes run and commits to actions. When all actions are complete, the environment is updated based on all the actions of all the agents during that time interval. Because of the separation of agent and simulator knowledge, each agent's frame system is active during its processing time, and the simulator's frame system is active in the interval during which the environment is updated (the simulator interval). In order to handle the mental actions in the implemented examples described in Chapter 7, a separate process had to be implemented in the agent to carry them out before its time interval was up. This was required because mental actions serve only to modify the internals of the agent itself, and thus require access to the frames representing the agent's knowledge which are unavailable during the simulator interval. Mental actions are inherently different from physical actions, in that they do not directly affect the world, but still occupy time. Rather than having to implement this as an agent process, the simulator itself should be able to handle the effects of mental actions as well as physical actions. The difficulty in providing this facility is that it is more significantly affected by the general structure of the agent, and thus it is difficult to provide such a facility without introducing assumptions about this structure.

---

115 This is the reason that Section 6.4.5 recommends the use of anytime algorithms.

Finally, Gensim also requires the ability to support more complicated spatial representations than it currently allows. While a grid-based structure is used often in intelligent agent evaluation and testing, and Gensim's ability to use abstract named locations allows the implementation of irregular areas, more sophisticated reasoning is also often necessary. Describing the connections between many irregular areas (as is currently required) becomes tedious, limiting the applicability of the system in areas where large geographic regions must be represented, such as simulating intelligent agents in a natural resource management setting [Anderson and Evans, 1994].

9.4. Future Work

As Chapter 1 has described, intelligent agency is a field with connections to virtually all of artificial intelligence. Because of these connections, there is a great deal of future research that arises out of the Waffler architecture, both including and beyond the limitations discussed in Section 9.3.1.

Within the limitations previously described, the most significant to be addressed is the improvement of the implemented mechanisms for constraint representation and working memory. As described in Chapter 7, constraints were implemented procedurally, and working memory was not limited in terms of space. One of the most important initial research projects will be to modify the implemented agent used in the examples in Chapter 7 to more realistically simulate the agent's working memory, using a standardized representation for constraints. This will allow experimentation with an agent in the existing domain, controlling the conditions in the domain and exploring the effects of limiting the size of working memory on agent behaviour. This in turn will require a much larger repertoire of intentions than is currently available and will result in a more complete example and a better validation of the approach.

I also intend to experiment with constraint-directed control, both examining the effects of using more detailed representations of utility (e.g. splitting the simple utility factors used in the examples of Chapter 7 into the components of utility described in Section 5.3.4) and in exploring the use of behaviour modes (Section 5.4) in complex examples of improvisation. In conjunction with this and the research described above, I will also be examining the effects of varying memory management policies on agent behaviour. These will also require a much more complete domain and agent knowledge base than that provided in Chapter 7. Beyond these improvements, I intend to explore practical applications of improvisation, including its application to

automation in manufacturing and in modelling human improvisation for the purposes of human-computer interaction.

There a number of important areas of research connected to improvisation about which this dissertation raises important questions and which will be the subject of future research. The first is that of motivation and improvisation. As described in Section 9.3.1, motivation is an important component of improvisation and is generally poorly understood. Future research will entail development of a model of motivation based on emotion and social knowledge for the purposes of improvisation.

Another important area is that of reasoning about function. Significant research in this area has only begun to be performed in the past few years, and its connection to improvisation is very significant in terms of higher degrees of improvisation than have been emphasized in this dissertation. The examples described in Chapter 7 all employ reasoning about function to varying degrees, but this has only been implemented in a primitive manner. For example, Section 7.3.3 shows an improvising agent reasoning about the function of a stovetop kettle and making a substitution. Improvisation often involves much higher degrees of reasoning about function than this: for example, I may have to reach a high shelf and recognize that I can use a chair to stand on, irrespective of the fact that that is not the primary purpose of a chair. I believe that constraint-directed reasoning can also be employed in reasoning about function at these more complex levels: Constraints should be able describe the features of the application, which can in turn act as reminders into a long-term memory of objects connected to this function. When overconstrained, however, constraints can also be relaxed, allowing more satisficing substitutions.

I also intend to develop a model of cooperative improvisation in multi-agent situations by combining the model presented in this dissertation and that of Evans et al. [1992] described in Section 8.5. This will allow much more sophisticated environments for improvisation, and will support more flexible cooperative behaviour, something that has been difficult to achieve in the past.

Finally, there is significant potential for research involving the creation of routines out of episodes of improvised behaviour. This is a much more extended project than those described above, but also one with much more significance to the field of AI. This will involve research into generalization and the connections between episodic and semantic memory in improvisation.

Gensim also has much potential for future research. As mentioned in Section 9.3.2, the flexibility of Gensim is paid for through much additional work in defining a domain and the agents that inhabit it. While this is unavoidable, there is much that can be done to improve the ease of use of the system, such as a more sophisticated programming interface for the system, and a graphical user interface. Currently, textual output informing the user of the domain changes that occur on each cycle, the actions taken by each agent, and the perceptual information given to each agent is the limit of Gensim's capabilities. A graphical user interface will allow Gensim to demonstrate the behaviour of agents in a particular environment in real time, as opposed to forcing users to pour over printed transcripts.

The separation of agent and environment knowledge supported by Gensim also shows great potential for future work on the agent's relationship to its environment and how agents actively structure the world around themselves for convenience in their activities. Hammond and Converse [1991] have begun to examine the ways in which intelligent agents make the world around themselves more stable, in order to make planning easier (for example, having standardized storage places or policies for activity). These concepts are intimately related to supporting improvisation (i.e. in providing regularities in the in environment for the agent to take advantage of). Hammond and Converse have concentrated on defining the types of stability an agent can actively bring to its environment, rather than the mechanisms by which this takes place. In applying a routine to a less-than-familiar situation (e.g. making tea at a neighbour's house or in a hotel), there are many cases where the agent can structure its environment to more closely approximate its usual routine. These situations are especially relevant to this area, and will be the subject of future work.

I also intent to work on making Gensim more useful as a simulation tool for other fields. For example, I have been exploring the needs necessary for the system to be used in a natural resource management context, providing accurate simulation of domains that include intelligent agents, rather than simply providing a testbed for the agents themselves [Anderson and Evans, 1994]. The first step in this direction will be the improvement of Gensim's spatial representation abilities, as described in Section 9.3.2. For this purpose as well as to aid in dealing with some of Gensim's other limitations (e.g. non-transparent time-sharing), I also intend to port the system to a broader hardware platform.

9.5. Summary

This Chapter has discussed the manner in which AI systems are and should be evaluated, and has recounted the contributions and limitations of the work described in this dissertation. As should occur in all scientific research, I have attempted as much as possible to emphasize the limitations of this work as well as its contributions, in the hope that this will encourage future research on questions that arise from this research.

I have also attempted to illustrate throughout this dissertation the central role of everyday activities in an understanding of human cognition, the role of other areas such as perception and representation in an architecture for everyday activities, and the potential benefits that a more thorough understanding of everyday activities will bring to AI. Research in such central, well-connected areas is important, because it encourages a holistic viewpoint that has been lacking in AI to date. The quote taken from [Dehn and Schank,1982] at the beginning of this Chapter has thus far been largely the case: AI has been a collection of sub-specialties, and when a significant degree of understanding has developed in a particular area, that area becomes less associated with AI and more associated with applications-oriented computer science. Greater research on the more central issues of intelligence and in the development of approaches that cross the boundaries between sub-specialties in AI will serve to unify the field and will be more productive in the long term.

APPENDIX A

EXAMPLE ONE OUTPUT

```
Locations of domain objects:
  Agent1      : (1 3)
  The-Phone   : (3 3)
  The-Spoon   : (1 3)
  The-Cupboard : (1 1)
  The-Tea     : Nil
  Counter1    : (1 3)
  Counter2    : (3 1)
  The-Sugar-Jar : (1 3)
  The-Cup     : (1 3)
Pending Events: ((1 (T95 Ring))) ;phone will ring in one cycle.

Running process P1 of agent Agent1 for 2.0 seconds

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Add-Sugar1
Going through steps looking for recommendations...
Step S8 recommends Adopt intention to Clean-Up The-Spoon with a utility of 10
Raw utility is 10, modifying by overall importance of 6
New Recommendation:
```

Adopt intention to Clean-Up The-Spoon recommended with utility 8 made by Add-Sugar1

Step S9 recommends Adopt intention to Clean-Up The-Sugar-Jar with a utility of 10
Raw utility is 10, modifying by overall importance of 6
New Recommendation:
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1

Processing constraints...
Processing concepts...
processing constraint *Check-For-Messes* from The-Spoon
recognized a new instance of Messes. Designating it Messes511
Adding concept Messes511 to working memory
processing constraint *Clean-Up-Messes* from Messes511
This constraint is activated and recommends: Adopt intention to Clean-Up Messes511 with utility 8
This recommendation actually refers to The-Spoon
This recommendation already made...modifying utility using 8
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1
Adopt intention to Clean-Up The-Spoon recommended with utility 9 made by Add-Sugar1

Committing to most appropriate action:
Adopt intention to Clean-Up The-Spoon recommended with utility 9 made by Add-Sugar1

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Add-Sugar1 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Clean-Up The-Spoon
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
The Phone Rings....
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
Adding concept Counter1 to working memory
I see a Container
This object is The-Sugar-Jar in my knowledge base
I see a Cup
This object is The-Cup in my knowledge base
I hear a Phone Ring at location (3 3)
The Ringing Phone sets off a trigger on the Kitchen concept
Recalling concept The-Phone
Adding concept The-Phone to working memory

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Add-Sugar1
Going through steps looking for recommendations...
Step S9 recommends Adopt intention to Clean-Up The-Sugar-Jar with a utility of 10
Raw utility is 10, modifying by overall importance of 6
New Recommendation:
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1

processing intention: Clean-Up511
Going through steps looking for recommendations...
Step S1 recommends Adopt intention to Travel-To-Object The-Spoon with a utility of 10
Raw utility is 10, modifying by overall importance of 3
New Recommendation:
Adopt intention to Travel-To-Object The-Spoon recommended with utility 5 made by Clean-Up511

Processing constraints...
Processing concepts...
processing constraint *Check-For-Messes* from The-Spoon
processing constraint *Clean-Up-Messes* from Messes511
processing constraint *Answer-The-Phone* from The-Phone
This constraint is activated and recommends: Adopt intention to Answer-Phone The-Phone with utility 10
New Recommendation:

Adopt intention to Answer-Phone The-Phone recommended with utility 10 made by A-Constraint

Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1
Adopt intention to Travel-To-Object The-Spoon recommended with utility 5 made by Clean-Up511
Adopt intention to Answer-Phone The-Phone recommended with utility 10 made by A-Constraint

Committing to most appropriate action:
Adopt intention to Answer-Phone The-Phone recommended with utility 10 made by A-Constraint

Running process P4 of agent Agent1 for 2.0 seconds

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Answer-Phone The-Phone
The importance of this new intention is 10, the default
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
I see a Container
This object is The-Sugar-Jar in my knowledge base
I see a Cup
This object is The-Cup in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...

processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Add-Sugar1
Going through steps looking for recommendations...
Step S9 recommends Adopt intention to Clean-Up The-Sugar-Jar with a utility of 10
Raw utility is 10, modifying by overall importance of 6
New Recommendation:
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1

processing intention: Clean-Up511
Going through steps looking for recommendations...
Step S1 recommends Adopt intention to Travel-To-Object The-Spoon with a utility of 10
Raw utility is 10, modifying by overall importance of 3
New Recommendation:
Adopt intention to Travel-To-Object The-Spoon recommended with utility 5 made by Clean-Up511

processing intention: Answer-Phone511
Going through steps looking for recommendations...
Step S1 recommends Adopt intention to Travel-To-Object The-Phone with a utility of 10
Raw utility is 10, modifying by overall importance of 10
New Recommendation:
Adopt intention to Travel-To-Object The-Phone recommended with utility 10 made by Answer-Phone511

Processing constraints...
Processing concepts...
processing constraint *Clean-Up-Messes* from Messes511
processing constraint *Answer-The-Phone* from The-Phone
processing constraint *Check-For-Messes* from The-Spoon
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Adopt intention to Clean-Up The-Sugar-Jar recommended with utility 8 made by Add-Sugar1
Adopt intention to Travel-To-Object The-Spoon recommended with utility 5 made by Clean-Up511
Adopt intention to Travel-To-Object The-Phone recommended with utility 10 made by Answer-Phone511

Committing to most appropriate action:
Adopt intention to Travel-To-Object The-Phone recommended with utility 10 made by Answer-Phone511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Forgetting intention Add-Sugar1...
Making sure to forget subintentions too...
Forgetting intention Clean-Up511...
Making sure to forget subintentions too...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...
Age causes concept Self to fade from working memory...
Removing concept Self from working memory
Age causes concept Messes511 to fade from working memory...
Removing concept Messes511 from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Travel-To-Object The-Phone
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
Adding concept Self to working memory
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
I see a Container
This object is The-Sugar-Jar in my knowledge base
I see a Cup
This object is The-Cup in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9

Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...
processing intention: Travel-To-Object511
Calling generator to get recommendations...
New Recommendation:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Processing constraints...
Processing concepts...
processing constraint *Check-For-Messes* from The-Spoon
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Committing to most appropriate action:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Travel-To-Object511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Action recorded: Move (S)
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Move S)
Action carried out by Agent1: (Move S)
New location: (2 3)

```
Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==============================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
I see a Container
This object is The-Sugar-Jar in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...
processing intention: Travel-To-Object511
Calling generator to get recommendations...
New Recommendation:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Processing constraints...
Processing concepts...
processing constraint *Check-For-Messes* from The-Spoon
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Committing to most appropriate action:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Running process P4 of agent Agent1 for 2.0 seconds
```

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Travel-To-Object511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...
Age causes concept Counter1 to fade from working memory...
Removing concept Counter1 from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Action recorded: Move (S)
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Move S)
Action carried out by Agent1: (Move S)
New location: (3 3)

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...

processing intention: Travel-To-Object511
Calling generator to get recommendations...
New Recommendation:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Processing constraints...
Processing concepts...
processing constraint *Check-For-Messes* from The-Spoon
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Committing to most appropriate action:
Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Travel-To-Object511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...
Age causes concept The-Sugar-Jar to fade from working memory...
Removing concept The-Sugar-Jar from working memory
Age causes concept The-Spoon to fade from working memory...
Removing concept The-Spoon from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Travel-To The-Phone recommended with utility 10 made by Travel-To-Object511

Completed intention Travel-To-Object511
updating parent intention to say this step is complete...
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...
Step S2 recommends Adopt intention to Talk-On-Phone The-Phone with a utility of 10
Raw utility is 10, modifying by overall importance of 10
New Recommendation:
Adopt intention to Talk-On-Phone The-Phone recommended with utility 10 made by Answer-Phone511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Adopt intention to Talk-On-Phone The-Phone recommended with utility 10 made by Answer-Phone511

Committing to most appropriate action:
Adopt intention to Talk-On-Phone The-Phone recommended with utility 10 made by Answer-Phone511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...
Age causes concept Self to fade from working memory...

```
Removing concept Self from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Talk-On-Phone The-Phone
Sensory request recorded: Look-Around

==============================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==============================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
Adding concept Self to working memory
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...
processing intention: Talk-On-Phone511
Calling generator to get recommendations...
New Recommendation:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511
```

Committing to most appropriate action:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Talk-On-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Action recorded: Chat
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Chat)
Action carried out by Agent1: (Chat)

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511

Going through steps looking for recommendations...
processing intention: Talk-On-Phone511
Calling generator to get recommendations...
New Recommendation:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Committing to most appropriate action:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Talk-On-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Action recorded: Chat
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Chat)
Action carried out by Agent1: (Chat)

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent

This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

processing intention: Answer-Phone511
Going through steps looking for recommendations...
processing intention: Talk-On-Phone511
Calling generator to get recommendations...
New Recommendation:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Committing to most appropriate action:
Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Answer-Phone511 and its sub-intentions are immune...
Intention Have-Tea1 not forgotten - Importance is 7, must be less than 7 to forget...
Intention Answer-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Intention Talk-On-Phone511 is immune from forgetting via its participation in Answer-Phone511...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Chat-On The-Phone recommended with utility 10 made by Talk-On-Phone511

Completed intention Talk-On-Phone511

```
updating parent intention to say this step is complete...
Parent intention is also complete...
Completed intention Answer-Phone511
updating parent intention to say this step is complete...
No parent for this intention
Sensory request recorded: Look-Around

==============================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==============================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
Step S2 recommends Adopt intention to Drink The-Tea with a utility of 9
Raw utility is 9, modifying by overall importance of 7
New Recommendation:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

Committing to most appropriate action:
Adopt intention to Drink The-Tea recommended with utility 7 made by Have-Tea1

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
```

```
Causing unused concepts to fade from working memory...
Age causes concept Self to fade from working memory...
Removing concept Self from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Drink The-Tea
Sensory request recorded: Look-Around

==============================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==============================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
Adding concept Self to working memory
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
processing intention: Drink511
Going through steps looking for recommendations...
Step S1 recommends Adopt intention to Travel-To-Object The-Tea with a utility of Nil
New Recommendation:
Adopt intention to Travel-To-Object The-Tea recommended with utility 7 made by Drink511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Travel-To-Object The-Tea recommended with utility 7 made by Drink511

Committing to most appropriate action:
Adopt intention to Travel-To-Object The-Tea recommended with utility 7 made by Drink511

Running process P4 of agent Agent1 for 2.0 seconds
```

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Drink511 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Travel-To-Object The-Tea
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
processing intention: Drink511
Going through steps looking for recommendations...
processing intention: Travel-To-Object511
Calling generator to get recommendations...
New Recommendation:
Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511

Committing to most appropriate action:
Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Drink511 is immune from forgetting via its participation in Have-Tea1...
Intention Travel-To-Object511 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...
Age causes concept The-Phone to fade from working memory...
Removing concept The-Phone from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511

Action recorded: Move (N)
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Move N)
Action carried out by Agent1: (Move N)
New location: (2 3)

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Phone
This object is The-Phone in my knowledge base
Adding concept The-Phone to working memory
I see a Spoon
This object is The-Spoon in my knowledge base
Adding concept The-Spoon to working memory
The-Spoon participates in my Messes511 concept
Adding concept Messes511 to working memory
I see a Work-Surface
This object is Counter1 in my knowledge base
Adding concept Counter1 to working memory
I see a Container
This object is The-Sugar-Jar in my knowledge base
Adding concept The-Sugar-Jar to working memory

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...

processing intention: Have-Tea1
Going through steps looking for recommendations...
processing intention: Drink511
Going through steps looking for recommendations...
processing intention: Travel-To-Object511
Calling generator to get recommendations...
New Recommendation:
Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511

Processing constraints...
Processing concepts...
processing constraint *Answer-The-Phone* from The-Phone
processing constraint *Check-For-Messes* from The-Spoon
processing constraint *Clean-Up-Messes* from Messes511
This constraint is activated and recommends: Adopt intention to Clean-Up Messes511 with utility 8
This recommendation actually refers to The-Spoon
New Recommendation:
Adopt intention to Clean-Up The-Spoon recommended with utility 8 made by A-Constraint

Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Travel-To The-Tea recommended with utility 7 made by Travel-To-Object511
Adopt intention to Clean-Up The-Spoon recommended with utility 8 made by A-Constraint

Committing to most appropriate action:
Adopt intention to Clean-Up The-Spoon recommended with utility 8 made by A-Constraint

Running process P4 of agent Agent1 for 2.0 seconds

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Clean-Up The-Spoon
The importance of this new intention is 5, the default
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========
End of example...halting simulator
A halt condition was satisfied
Simulation ends.

APPENDIX B

EXAMPLE TWO OUTPUT

```
Locations of domain objects:
  Agent1      : (1 3)
  The-Phone   : (3 3)
  The-Spoon   : (1 3)
  The-Cupboard : (1 1)
  Counter1    : (1 3)
  Counter2    : (3 1)
  The-Sugar-Jar : (1 3)
  The-Cup     : (1 3)

Running process P1 of agent Agent1 for 2.0 seconds

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Step S1 recommends Adopt intention to Boil-Water  with a utility of 8
Raw utility is 8, modifying by overall importance of 7
New Recommendation:
Adopt intention to Boil-Water  recommended with utility 6 made by Have-Tea1

Applying constraints on step S2
Processing constraints...
Processing concepts...
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Boil-Water  recommended with utility 6 made by Have-Tea1
```

Committing to most appropriate action:
Adopt intention to Boil-Water recommended with utility 6 made by Have-Tea1

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Boil-Water
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
Adding concept Counter1 to working memory
I see a Container
This object is The-Sugar-Jar in my knowledge base
I see a Cup
This object is The-Cup in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
processing intention: Boil-Water394
Going through steps looking for recommendations...
Applying constraints on step S1
Attempting to bind 'Boiling-Container
The following default is recommended automatically via preference:
New Recommendation:

Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water394

The following alternatives arise immediately from objects in working memory...
Recalling purpose concept Boiling-Container to working memory...
Adding concept Boiling-Container to working memory
Processing constraints...
Processing concepts...
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water394
Action Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water394
cannot be adopted...limit on utility from that intention is 7

There are no more actions to select from...

Running process P4 of agent Agent1 for 2.0 seconds

Running process P5 of agent Agent1 for 2.0 seconds

No action to commit to: doing nothing
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
==========

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Spoon
This object is The-Spoon in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
I see a Container
This object is The-Sugar-Jar in my knowledge base
I see a Cup
This object is The-Cup in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
processing intention: Boil-Water394
Going through steps looking for recommendations...
Applying constraints on step S1
Attempting to bind 'Boiling-Container
The following default is recommended automatically via preference:
New Recommendation:
Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water394

The following alternatives arise immediately from objects in working memory...
New Recommendation:
Adopt intention to Find-A Pot recommended with utility 8 made by A-Constraint

New Recommendation:
Adopt intention to Find-A Electric-Kettle recommended with utility 9 made by A-Constraint

This recommendation already made...modifying utility using 4
Recalling purpose concept Boiling-Container to working memory...
Processing constraints...
Processing concepts...
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Find-A Pot recommended with utility 8 made by A-Constraint
Adopt intention to Find-A Electric-Kettle recommended with utility 9 made by A-Constraint
Adopt intention to Find-A Stovetop-Kettle recommended with utility 5 made by Boil-Water394
Action Adopt intention to Find-A Stovetop-Kettle recommended with utility 5 made by Boil-Water394
cannot be adopted...limit on utility from that intention is 7

Committing to most appropriate action:
Adopt intention to Find-A Electric-Kettle recommended with utility 9 made by A-Constraint

Running process P4 of agent Agent1 for 2.0 seconds

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Find-A Electric-Kettle
The importance of this new intention is 7, the default
Sensory request recorded: Look-Around

==========
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

```
Processing sensory requests...
Processing Agent1's sensory request: Look-Around
============================================================
End of example...halting simulator
A halt condition was satisfied
Simulation ends.
```

APPENDIX C

EXAMPLE THREE OUTPUT

```
Locations of domain objects:
  Agent1      : (4 1)
  The-Phone   : (3 3)
  The-Spoon   : (1 3)
  The-Cupboard : (1 1)
   The-Electric-Kettle: (2 1)
  Counter1    : (1 3)
  Counter2    : (3 1)
   The-Sugar-Jar : (1 3)
  The-Cup     : (1 3)
  The-Sink    : (1 2)
  The-Stove   : (2 1)

Running process P1 of agent Agent1 for 2.0 seconds

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
processing intention: Boil-Water1
Going through steps looking for recommendations...
Applying constraints on step S1
Attempting to bind 'Boiling-Container
The following default is recommended automatically via preference:
New Recommendation:
Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water1

The following alternatives arise immediately from objects in working memory...
Recalling purpose concept Boiling-Container to working memory...
```

```
Adding concept Boiling-Container to working memory
Processing constraints...
Processing concepts...
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water1

Committing to most appropriate action:
Adopt intention to Find-A Stovetop-Kettle recommended with utility 4 made by Boil-Water1

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Boil-Water1 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Find-A Stovetop-Kettle
Adding concept Stovetop-Kettle to working memory
Sensory request recorded: Look-Around

================================================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
================================================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Work-Surface
This object is Counter1 in my knowledge base
Adding concept Counter1 to working memory

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
processing intention: Boil-Water1
Going through steps looking for recommendations...
```

```
processing intention: Find-A412
Calling generator to get recommendations...
New Recommendation:
Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412

Processing constraints...
Processing concepts...
Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412

Committing to most appropriate action:
Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Boil-Water1 is immune from forgetting via its participation in Have-Tea1...
Intention Find-A412 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...

Running process P5 of agent Agent1 for 2.0 seconds

Committing to: Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412

Action recorded: Move (N)
Sensory request recorded: Look-Around

================================================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities
processing action (Move N)
Action carried out by Agent1: (Move
                    N)
New location: (3 1)

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
================================================================

Running process P1 of agent Agent1 for 2.0 seconds

I see a Primary-Agent
This object is Self in my knowledge base
I see a Electric-Kettle
This object is The-Electric-Kettle in my knowledge base
```

Adding concept The-Electric-Kettle to working memory
this matches the purpose of intention Find-A412 and is a valid alternative...
I see a Work-Surface
This object is Counter1 in my knowledge base
I see a Stove
This object is The-Stove in my knowledge base

Running process P2 of agent Agent1 for 2.0 seconds

Processing intentions...
processing intention: Have-Tea1
Going through steps looking for recommendations...
Applying constraints on step S2
processing intention: Boil-Water1
Going through steps looking for recommendations...
processing intention: Find-A412
Calling generator to get recommendations...
New Recommendation:
Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412

Processing constraints...
Processing concepts...
Activating constraint *Use-Whats-Convenient*
purpose of The-Electric-Kettle allows it to be an alternative to Find-A412
New Recommendation:
Adopt intention to Find-A The-Electric-Kettle recommended with utility 10 made by Boil-Water1

Processing settings...

Running process P3 of agent Agent1 for 2.0 seconds

Potential actions:
Look-For-A Stovetop-Kettle recommended with utility 5 made by Find-A412
Adopt intention to Find-A The-Electric-Kettle recommended with utility 10 made by Boil-Water1

Committing to most appropriate action:
Adopt intention to Find-A The-Electric-Kettle recommended with utility 10 made by Boil-Water1

Running process P4 of agent Agent1 for 2.0 seconds

Causing unused intentions to fade from working memory...
intention Have-Tea1 and its sub-intentions are immune...
Intention Have-Tea1 is immune from forgetting via its participation in Have-Tea1...
Intention Boil-Water1 is immune from forgetting via its participation in Have-Tea1...
Intention Find-A412 is immune from forgetting via its participation in Have-Tea1...
Causing unused concepts to fade from working memory...
Age causes concept Self to fade from working memory...
Removing concept Self from working memory
Age causes concept The-Spoon to fade from working memory...

```
Removing concept The-Spoon from working memory
Age causes concept The-Sugar-Jar to fade from working memory...
Removing concept The-Sugar-Jar from working memory
Age causes concept The-Stove to fade from working memory...
Removing concept The-Stove from working memory

Running process P5 of agent Agent1 for 2.0 seconds

Adopting intention: Find-A The-Electric-Kettle
intention Find-A412 has been adopted for the same reason...
Activating constraint *Only-One-Object-For-Role*
abandoning the poorer of Find-A412 and Find-A413
Find-A412 is a better alternative than Find-A413...
abandoning intention Find-A413 in favor of the better alternative...
Sensory request recorded: Look-Around

============================================================
Updating the environment:
allowing scheduled events to occur...
Checking for Agent activities

Processing sensory requests...
Processing Agent1's sensory request: Look-Around
============================================================
End of example...halting simulator
A halt condition was satisfied
Simulation ends.
```

BIBLIOGRAPHY

[Ackerman, 1992] Ackerman, Philip L., "Human Intelligence", in Schapiro, Stuart C. (Ed.), *The Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley and Sons), Vol. 1, 1992, pp. 706-715.

[Ackoff, 1981] Ackoff, Russell L., "The Art and Science of Mess Management", *Interfaces*, Vol. 11, No. 1, 1981, pp. 20-26.

[Agre, 1988] Agre, Philip E., *The Dynamic Structure of Everyday Life*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (AI Technical Report 1085, MIT AI Lab), 1988. 282 pp.

[Agre and Chapman, 1987] Agre, Philip E. and David Chapman, "Pengi: An Implementation of a Theory of Activity", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987, pp. 196-201.

[Agre and Chapman, 1989] Agre, Philip E. and David Chapman, *What are Plans For?*, Technical Report 1050, MIT AI Lab, 1989. 29 pp.

[Agre and Horswill, 1992] Agre, Philip E., and Ian D. Horswill, "Cultural Support for Improvisation", *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992, pp. 363-368.

[Alterman, 1988] Alterman, Richard, "Adaptive Planning", *Cognitive Science*, Vol. 12, 1988, pp. 393-421.

[Ambros-Ingerson, 1987] Ambros-Ingerson, Jose A., *IPEM: Integrated Planning, Execution, and Monitoring*, M. Phil. Dissertation, Department of Electrical Engineering and Computer Science, University of Essex, 1987. 106 pp.

[Anderson, 1991] Anderson, John E., *Plan Use in Everyday Activity*, Technical Report 91-03, Department of Computer Science, University of Manitoba, 1991. 38 pp.

[Anderson and Evans, 1991] Anderson, John E., and Mark Evans, "An Architecture for Reactive and Strategic Planning", in *Proceedings of the Fourth UNB AI Symposium*, Fredericton, NB, 1991, pp. 195-207.

[Anderson and Evans, 1993] Anderson, John E., and Mark Evans, "Supporting Flexible Autonomy in a Simulation Environment for Intelligent Agent Designs", *Proceedings of the Fourth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Tuscon, AZ, 1993, pp. 60-66.

[Anderson and Evans, 1994] Anderson, John E., and Mark Evans, "Intelligent Agent Modelling for Natural Resource Management", *International Journal of Mathematical and Computer Modelling*, Vol. 20, No. 8, October, 1994, pp. 109-119.

[Ashcraft, 1989] Ashcraft, Mark H., *Human Memory and Cognition* (Glenview, IL: Scott, Foresman, and Company), 1989. 770 pp.

[Atkinson, 1992] Atkinson, Brooks (Ed.), *Walden and Other Writings of Henry David Thoreau* (New York: Modern Library), 1992. 769 pp.

[Aurelias, 1905] Marcus Aurelias, *The Twelve Books of the Emperor Marcus Aurelias Antoninus* (London: Arthur L. Humphreys), 1905. 288 pp.

[Austin, 1970] Austin, John Langshaw, *Philosophical Papers* (London: Oxford University Press), Second Edition, 1970. 287 pp.

[Barker, 1968] Barker, Roger G., *Ecological Psychology: Concepts and Methods for Studying the Environment of Human Behaviour* (Stanford, CA: Stanford University Press), 1968. 242 pp.

[Barker et al., 1978] Barker, Roger G., and Associates, *Habitats, Environments, and Human Behaviour* (San Francisco: Jossey-Bass Publishers), 1978. 327 pp.

[Barker et al., 1992] Barker, Ken, Mark Evans, and John E. Anderson, "Quantification of Autonomy in Multi-Agent Systems", *Proceedings of the AAAI Workshop on Cooperation Among Heterogeneous Intelligent Systems*, San Jose, CA, 1992 (also Technical Report 92-08, Department of Computer Science, University of Manitoba). 6 pp.

[Barr and Feigenbaum, 1981] Barr, Avron, and Edward A. Feigenbaum (Eds)., *The Handbook of Artificial Intelligence*, Volume I (Los Altos, CA: William Kaufmann), 1981. 409 pp.

[Barrett, 1962] Barrett, William, *Irrational Man: A Study in Existential Philosophy* (Garden City, NY: Doubleday), 1962. 314 pp.

[Bartlett, 1958] Bartlett, Sir Frederic C., *Thinking: An Experimental and Social Study* (London: Unwin University Books), 1958. 203 pp.

[Bartlett, 1967] Bartlett, Sir Frederic C., *Remembering: A Study in Experimental and Social Psychology* (London: Cambridge University Press), 1967. 317 pp.

[Baykan and Fox, 1987] Baykan, Carl A., and Mark S. Fox, "An Investigation of Opportunistic Constraint Satisfaction in Space Planning", *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, pp. 1035-1038.

[Birch, 1945] Birch, H. G., "The Relation of Previous Experience to Insightful Problem-solving", *Journal of Comparative Psychology*, Vol. 38, 1945, pp. 367-383.

[Birtwhistle, 1973] Birtwhistle, Graham, Ole-Johan Dahl, Bjørn Myhrhaug, and Kristen Nygaard, *Simula Begin* (Lund, Sweden: Studentlitteratur), 1973. 391 pp.

[Boden, 1973] Boden, Margaret, "The Structure of Intentions", *Journal for the Study of Social Behaviour*, Vol 3., No. 1, April, 1973, pp. 23-46.

[Bond and Gasser, 1988] Bond, Alan H., and Les Gasser, *Readings in Distributed Artificial Intelligence* (San Mateo, CA: Morgan Kaufmann), 1988. 649 pp.

[Bratman, 1987] Bratman, Michael E., *Intentions, Plans, and Practical Reason* (Cambridge, MA: Harvard), 1987. 200 pp.

[Bratman et al., 1988] Bratman, Michael E., David J. Israel, and Martha E. Pollack, *Plans and Resource-Bounded Practical Reasoning*, Technical Note 425R, SRI International, 1988. 28 pp. (also available excerpted in *Computational Intelligence*, Vol. 4, 1988, pp. 349-355).

[Brooks, 1986] Brooks, Rodney, A., "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. RA-2, April, 1986, pp. 14-23.

[Brooks, 1991] Brooks, Rodney A., "Elephants Don't Play Chess", in Maes, Pattie (Ed.), *Designing Autonomous Agents* (Cambridge, MA: MIT Press), 1991, pp. 3-15.

[Buchanan and Smith, 1989] Buchanan, Bruce G., and Reid G. Smith, "Fundamentals of Expert Systems", in Barr, Avron, Paul R. Cohen, and Edward A. Feigenbaum (Eds.), *The Handbook of AI*, Volume IV (Reading, MA: Addison-Wesley), 1989, pp. 149-192.

[Bylander, 1992] Bylander, Tom, "Complexity Results for Extended Planning", *Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, June, 1992, pp. 20-27.

[Cailliet et al., 1971] Cailliet, Greg, Paulette Setzer, and Milton Love, *Everyman's Guide to Ecological Living* (New York: MacMillan), 1971. 119 pp.

[Campbell, 1980] Campbell, Susan, *Cook's Tools: The Complete Manual of Kitchen Implements and How to Use Them* (Toronto, ON: Bantam), 1980. 288 pp.

[Chapman, 1987] Chapman, David, Planning for Conjunctive Goals, *Artificial Intelligence*, Vol. 32, 1987, pp. 333-378.

[Chapman, 1990] Chapman, David, *Vision, Instruction, and Action*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (AI Technical Report 1204, MIT AI Lab), 1990, 244 pp.

[Coenders, 1992] Coenders, A. *The Chemistry of Cooking: an Account of What Happens to Food Before, During, and After Cooking* (Carnforth, UK: Parthenon Publishing Group), 1992. 276 pp.

[Cohen and Feigenbaum, 1982] Cohen, Paul R., and Edward A. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence*, Volume III (Los Altos, CA: William Kaufmann), 1982. 639 pp.

[Cohen and Perrault, 1979] Cohen, Philip R., and C. Raymond Perrault, "Elements of a Plan-Based Theory of Speech Acts", *Cognitive Science*, Vol. 3, No. 3, 1979, pp. 177-212.

[Cohen et al., 1989] Cohen, Paul R., Michael L. Greenberg, David M. Hart, and Adele E. Howe, *Trial by Fire: Understanding the Design Requirements of Agents in Complex Environments*, COINS Technical Report 89-61, Department of Computer and Information Science, University of Massachusetts at Amherst, 1989. 22 pp. (also *AI Magazine*, Vol. 10, No. 3, Fall, 1989, pp. 34-48).

[Collinot and Le Pape, 1991] Collinot, Anne, and Claude Le Pape, "Adapting the Behaviour of a Job-shop Scheduling System", *Decision Support Systems,* Vol. 7, 1991, pp. 341-353.

[Covigaru and Lindsay, 1991] Covigaru, Arie A., and Robert K. Lindsay, "Deterministic Autonomous Systems", *AI Magazine,* Vol. 12, No. 3, Fall, 1991, pp. 110-117.

[Cutler, 1985] Cutler, Paul, *Problem-solving in Clinical Medicine: from Data to Diagnosis* (Baltimore, MD: Williams & Wilkins), 1985. 603 pp.

[Davis and Blumenthal, 1991] Davis, Paul K., and Donald Blumenthal, *The Base of Sand Problem: A White Paper on the State of Military Combat Modelling,* Rand Note N-3148-OSD/DARPA, The Rand Corporation, 1991. 46 pp.

[Davis, 1991] Davis, Randall, "A Tale of Two Knowledge Servers", *AI Magazine,* Vol. 12 No. 3, Fall, 1991, pp. 118-120.

[Davis et al., 1977] Davis, Randall, Bruce G. Buchanan, and Edward H. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program", *Artificial Intelligence,* Vol. 8, 1977, pp. 15-45.

[Dean and Boddy, 1988] Dean, Thomas, and Mark Boddy, "An Analysis of Time-Dependent Planning", *Proceedings of the Seventh National Conference on Artificial Intelligence,* St. Paul, MN, 1988, pp. 49-54.

[Dean et al., 1993] Dean, Thomas, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson, "Planning with Deadlines in Stochastic Domains", *Proceedings of the Eleventh National Conference on Artificial Intelligence,* Washington, DC, 1993, pp. 574-579.

[Decker et al., 1989] Decker, Kieth S., Edmund H. Durfee, and Victor R. Lesser, "Evaluating Research in Cooperative Distributed Problem-solving", in Gasser, Les, and Michael N. Huhns (Eds.), *Distributed Artificial Intelligence,* Volume II (San Mateo, CA: Morgan Kaufmann), 1989, pp. 485-519.

[Dehn and Shank, 1982] Dehn, Natalie, and Roger Schank, "Artificial and Human Intelligence", in Sternberg, Robert J. (Ed.), *Handbook of Human Intelligence* (London: Cambridge University Press), 1982, pp. 352-591.

[Desimone, 1992] Desimone, Roberto, "Socap: Lessons Learned in Applying SIPE-2 to the Military Operations Crisis Action Planning Domain", *Proceedings of the Spring Symposium on Practical Approaches to Scheduling and Planning,* AAAI Technical Report SS-92-01, 1992, pp. 156-159.

[Donagan, 1987] Donagan, Alan, *Choice: The Essential Element in Human Action* (London: Routledge and Kegan Paul), 1987. 197 pp.

[Dreyfus, 1981] Dreyfus, Hubert L., "From Micro-Worlds to Knowledge Representation: AI at an Impasse", in Haugeland, John (Ed.), *Mind Design* (Cambridge, MA: MIT Press), 1981, pp. 161-204.

[Durfee, 1992] Durfee, Edmund H., "What Your Computer Really Needs to Know, You Learned in Kindergarten", *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992, pp. 858-864.

[Durfee and Montgomery, 1989] Durfee, Edmund H., and Thomas A. Montgomery, "MICE: A Flexible Testbed for Intelligent Coordination Experiments", *Proceedings of the Ninth Distributed Artificial Intelligence Workshop*, Eastsound, WA, September, 1989, pp. 25-40.

[EmpImm, 1991] Ministry of Employment and Immigration Canada, *A Newcomer's Guide to Canada*, (Ottawa, ON: Ministry of Supply and Services Canada), 1991. 85 pp.

[Engelson and Bertani, 1992] Engelson, Sean P., and Niklas Bertani, *Ars Magna: The Abstract Robot Simulator Manual*, Department of Computer Science, Yale University, October 1992. 73 pp.

[Erman et al., 1980] Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, June, 1980, pp. 213-253.

[Ernst and Newell, 1969] Ernst, George, and Allen Newell, *GPS: A Case Study in Generality and Problem-Solving* (New York: Academic Press), 1969. 294 pp.

[Estes, 1982] Estes, William K., "Learning, Memory, and Intelligence", in Sternberg, Robert J. (Ed.), *Handbook of Human Intelligence* (London: Cambridge University Press), 1982, pp. 170-224.

[Etzioni, 1993] Etzioni, Oren, "Intelligence without Robots: A Reply to Brooks", *AI Magazine*, Vol. 14, No. 4, Winter, 1993, pp. 7-13.

[Etzioni and Segal, 1992] Etzioni, Oren, and Richard Segal, "Softbots as Testbeds for Machine Learning", *Proceedings of the Machine Learning Workshop at AI/GI/VI '92*, University of British Columbia, 1992, pp. v1-v8.

[Evans, 1988] Evans, Mark, *A Knowledge-Based Model of Distributed Problem-Solving*, Ph.D. Dissertation, Department of Computer Science, University of Manitoba, 1988. 263 pp.

[Evans and Anderson, 1989] Evans, Mark, and John E. Anderson, "A Constraint-Directed Architecture for Multi-Agent Planning", in *Proceedings of the Ninth AAAI Distributed Artificial Intelligence Workshop*, Eastsound, WA, 1989, pp. 1-24.

[Evans and Anderson, 1990] Evans, Mark, and John E. Anderson, "Constraint-Directed Intelligent Control in Multi-Agent Problem-solving", in Zeigler, Bernard, and Jerzy Rozenblit (Eds.), *AI, Simulation, and Planning in High Autonomy Systems*, (Los Alamitos, CA: IEEE Computer Society Press), 1990, pp. 42-40.

[Evans and Anderson, 1991] Evans, Mark, and John E. Anderson, *An Analysis of Constraints for Multi-Agent Problem-solving*, Technical Report 91-02, Department of Computer Science, University of Manitoba, January, 1991. 21 pp.

[Evans et al., 1992] Evans, Mark, John E. Anderson, and Geoff Crysdale, "Achieving Flexible Autonomy in Multi-Agent Systems using Constraints", *Applied Artificial Intelligence*, Vol. 6, No. 1, 1992, pp. 103-126.

[Fikes and Nilsson, 1971] Fikes, Richard E., and Nils J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem-solving", *Artificial Intelligence*, Vol. 2, 1971, pp. 189-208.

[Fikes et al., 1972] Fikes, Richard E., Peter E. Hart, and Nils J. Nilsson, "Learning and Executing Generalized Robot Plans", *Artificial Intelligence*, Vol. 3, 1972, pp. 251-288.

[Firby, 1987] Firby, R. James, "An Investigation into Reactive Planning in Complex Domains", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987, pp. 202-206.

[Firby, 1992] Firby, R. James, "Building Symbolic Primitives with Continuous Control Routines", *Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, 1992, pp. 62-69.

[Forbus, 1984] Forbus, Ken, "Qualitative Process Theory", *Artificial Intelligence*, Vol. 24, 1984, pp. 85-168.

[Fox, 1983] Fox, Mark S., *Constraint-Directed Search: A Case Study of Job Shop Scheduling*. Ph.D. Dissertation, School of Computer Science, Carnegie-Mellon University, 1983. 184 pp.

[Fox, 1987] Fox, Mark S., "Beyond the Knowledge Level", in Kerschberg, Larry (Ed.), *Expert Database Systems* (New York: Benjamin Cummings), 1987, pp 455-463.

[Fox, 1991] Fox, Mark S., *Personal Communication*, February, 1991.

[Fox and Nadel, 1989] Fox, Mark S., and Bernard Nadel, *Constraint-Directed Reasoning*, Tutorial Notes, Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, 1989. 130 pp.

[Fox and Kempf, 1985] Fox, Barry R. and Karl G. Kempf, "Opportunistic Scheduling for Robotic Assembly", *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Louis, MO, 1985, pp. 880-889.

[Friedland, 1977] Friedland, Edward I., "Values and Environmental Modelling", in Hall, Charles, and John Day (Eds.), *Ecosystem Modelling in Theory and Practice* (New York: John Wiley and Sons), 1977, pp. 115-132.

[Fulgham, 1986] Fulgham, Robert, *All I Really Need to Know I Learned in Kindergarten* (New York: Ivy Books), 1986. 196 pp.

[Gat, 1993] Gat, Erann, "On the Role of Stored Internal State in the Control of Autonomous Mobile Robots", *AI Magazine*, Vol. 14, No. 1, 1993, pp. 64-73.

[Gat et al., 1993] Gat, Erann, Mark Slack, Avi Kak, and Steve Chien, "Report on the AAAI Fall Symposium on Applications of AI to Real-World Autonomous Mobile Robots", *AI Magazine*, Vol. 14, No. 1, 1993, pp. 10-11.

[Genesereth and Nilsson, 1987] Genesereth, Michael R., and Nils J. Nilsson, *Logical Foundations of Artificial Intelligence* (Los Altos, CA: Morgan Kaufmann), 1987. 405 pp.

[Georgeff and Lansky, 1987] Georgeff, Michael P., and Amy L. Lansky, "Reactive Reasoning and Planning", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987, pp. 677-682.

[Ginsberg. 1989] Ginsberg, Matthew L., "Universal Planning: An (Almost) Universally Bad Idea", *AI Magazine*, Vol. 10, No. 4, 1989, pp. 40-44.

[Ginsberg, 1993] Ginsberg, Matthew L., *Essentials of Artificial Intelligence* (San Mateo, CA: Morgan Kaufmann), 1993. 430 pp.

[Goffman, 1974] Goffman, Erving, *Frame Analysis: An Essay on the Organization of Experience* (Cambridge, MA: Harvard University Press), 1974. 586 pp.

[Goldberg and Pohl, 1981] Goldberg, Allen, and Ira Pohl, "Is Complexity Theory of Use to AI?", in Eithorn, Alick, Ranan Banerji (Eds.), *Artificial and Human Intelligence* (Amsterdam: North-Holland), 1981, pp. 43-56.

[Goldstein, 1984] Goldstein, E. Bruce, *Sensation and Perception* (Belmont, CA: Wadsworth Publishing Co.), 1984. 481 pp.

[Gracián, 1992] Gracián, Baltazar, *The Art of Worldly Wisdom* (New York: Doubleday), 1992. Translated from the Spanish by Christopher Maurer, original 1642. 182 pp.

[Graham, 1965] Graham, Clarence H., "Perception of Movement", in Graham, Clarence H. (Ed.), *Vision and Visual Perception* (New York: Wiley), 1965, pp. 575-588.

[Green, 1969] Green, Cordell, "Application of Theorem Proving to Problem-solving", in *Proceedings of the First International Joint Conference on Artificial Intelligence,* Washington, DC, May, 1969, pp. 219-239.

[Guha and Lenat, 1990] Guha R. V., and Douglas B. Lenat, "CYC: A Midterm Report", *AI Magazine*, Vol. 11, No. 3, Fall, 1990, pp. 33-59.

[Hadwen, 1937] Hadwen, Sibylla, *Recipes for Serving 100* (Minneapolis, MN.: Burgess), 1937. 188 pp.

[Hammond, 1974] Hammond, John S. III, "Do's and Don'ts of Computer Models for Planning", *Harvard Business Review*, Vol. 52, pp. 110-123.

[Hammond, 1989a] Hammond, Kristian J., *Case-Based Planning: Viewing Planning as a Memory Task* (Boston, MA: Academic Press), 1989. 277 pp.

[Hammond, 1989b] Hammond, Kristian J., "Opportunistic Memory", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989, pp. 504-510.

[Hammond and Converse, 1991] Hammond, Kristian J., and Timothy M. Converse, "Stabilizing Environments to Facilitate Planning and Activity: An Engineering Argument", *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, 1991, pp. 787-793.

[Hammond et al., 1990] Hammond, Kristian J., Timothy Converse, and Charles Martin, "Integrating Planning and Acting in a Case-Based Framework", *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, 1990, pp. 292-297.

[Hanks et al., 1993] Hanks, Steve, Martha E. Pollack, and Paul R. Cohen, "Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures", *AI Magazine*, Vol. 14, No. 4, Winter, 1993, pp. 17-42.

[Hayes, 1985] Hayes, Patrick J., "Naive Physics I: Ontology for Liquids", in Hobbes, Jerry R. and Robert C. Moore (Eds.), *Formal Theories of the Commonsense World* (Norwood, NJ: Ablex), 1985, pp. 71-107.

[Hayes et al., 1994] Hayes, Patrick J., Kenneth M. Ford, and Neil Agnew, "On Babies and Bathwater: A Cautionary Tale", *AI Magazine*, Vol. 15, No. 4, Winter, 1994, pp. 15-26.

[Hayes-Roth, 1985] Hayes-Roth, Barbara, "A Blackboard Architecture for Control", *Artificial Intelligence*, Vol. 26, 1985, pp. 251-321.

[Hayes-Roth and Hayes-Roth, 1979] Hayes-Roth, Barbara, and Hayes-Roth, Frederick, "A Cognitive Model of Planning", *Cognitive Science*, Vol. 3, No. 4, 1979, pp. 275-310.

[Hayes-Roth, 1993] Hayes-Roth, Barbara, "Coordinating Control Modes with Control Situations, in *Foundations of Automatic Planning: The Classical Approach and Beyond*, AAAI Technical Report SS-93-03, 1993, pp. 44-47.

[Hayes-Roth et al., 1983] Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat, "An Overview of Expert Systems", in Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat (Eds.), *Building Expert Systems* (Reading, MA: Addison-Wesley), 1983, pp. 3-30.

[Hendler, 1992] Hendler, James (Ed.), *Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, 1992. 315 pp.

[Hobbs and Moore, 1985] Hobbs, Jerry R., and Robert C. Moore (Eds.), *Formal Theories of the Commonsense World* (Norwood, NJ: Ablex), 1985. 455 pp.

[Hobbs et al., 1985] Hobbs, Jerry R., Tom Blenko, Bill Croft, Greg Hager, Henry A. Kautz, Paul Kube, and Yoav Shoham, *Commonsense Summer: Final Report*, Report No. CSLI-85-35, Centre for the Study of Language and Information, Stanford University, 1985. 196 pp.

[Hodgson and Richards, 1974] Hodgson, John, and Ernest Richards, *Improvisation* (London: Eyre Methuen Ltd.), 1974. 209 pp.

[Hofstadter, 1985] Hofstadter, Douglas R., *Metamagical Themas: Questing for the Essence of Mind and Pattern* (New York: Basic Books), 1985. 852 pp.

[Howe, 1991] Howe, Adele, *Personal Communication*, August, 1991.

[Howe, 1993] Howe, Adele, "Evaluating Planning through Simulation: An Example Using Phoenix", *Proceedings of the workshop on the Foundations of Automatic Planning: The Classical Approach and Beyond*, AAAI Technical Report SS-93-03, 1993, pp. 53-57.

[Howe and Cohen, 1990] Howe, Adele, and Paul R. Cohen, *Responding to Environmental Change*, Technical Report 90-23, Department of Computer and Information Science, University of Massachusetts at Amherst, 1990. 12 pp.

[Howe and Cohen, 1991] Howe, Adele, and Paul R. Cohen, "Failure Recovery: A Model and Experiments", *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, 1991, pp. 801-808.

[Hullah, 1984] Hullah, Evelyn, *Cardinal's Handbook of Recipe Development* (Toronto, ON: Cardinal Kitchens), 1984. 159 pp.

[HumCog, 1984] Laboratory of Comparative Human Cognition, "Culture and Intelligence", in Sternberg, Robert J. (Ed.), *Handbook of Human Intelligence* (London: Cambridge University Press), 1982, pp. 642-692.

[Jackson, 1990] Jackson, Peter, *Introduction to Expert Systems* (Reading, MA: Addison-Wesley), 1990. 526 pp.

[Jencks and Silver, 1972] Jencks, Charles, and Nathan Silver, *Adhocism: The Case for Improvisation* (New York: Doubleday), 1972. 216 pp.

[Kaelbling and Rosenschein, 1991] Kaelbling, Leslie Pack, and Stan J. Rosenschein, "Action and Planning in Embedded Agents", in Maes, Pattie (Ed.), *Designing Autonomous Agents* (Cambridge, MA: MIT Press), 1991, pp. 35-48.

[Kinney and Georgeff, 1991] Kinney, David N., and Michael P. Georgeff, "Commitment and Effectiveness of Situated Agents", *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991, pp. 82-88.

[Kirsch and Maglio, 1992] Kirsch, David, and Paul Maglio, "Reaction and Reflection in Tetris", *Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, 1992, pp. 283-285.

[Krebsbach et al., 1992] Krebsbach, Kurt, Duane Olawsky, and Maria Gini, "An Empirical Study of Sensing and Defaulting in Planning", *Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, 1992, pp. 136-144.

[Kuipers, 1978] Kuipers, Benjamin, "Modelling Spatial Knowledge", *Cognitive Science*, Vol. 2, 1978, pp. 129-153.

[Kuipers, 1986] Kuipers, Benjamin, "Qualitative Simulation", *Artificial Intelligence*, Vol. 29, 1986, pp. 289-338.

[Kumar, 1992] Kumar, Vipin, "Algorithms for Constraint Satisfaction Problems: A Survey", *AI Magazine*, Vol. 13, No. 1, Spring, 1992, pp. 32-44.

[Laird et al., 1987] Laird, John E., Allen Newell, and Paul S. Rosenbloom, "SOAR: An Architecture for General Intelligence", *Artificial Intelligence*, Vol. 33, 1987, pp. 1-64.

[Lansky, 1986] Lansky, Amy, "A Representation of Parallel Activity", *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Timberline, OR, 1986, pp. 123-159.

[Lave et al., 1984] Lave, Jean, Michael Murtaugh, and Olivia de la Rocha, "The Dialectic of Arithmetic in Grocery Shopping", in Rogoff, Barbara and Jean Lave (Eds.), *Everyday Cognition: Its Development in Social Context* (Cambridge: Harvard University Press), 1984, pp. 67-94.

[Lenat et al., 1986] Lenat, Doug, M. Prakash, and M. Shepherd, "CYC: Using Common-Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks", *AI Magazine*, Vol. 6, No. 4., Winter, 1986, pp. 65-85.

[Le Pape, 1991] Le Pape, Claude, *Constraint Propagation in Planning and Scheduling*, CIFE Technical Report, Robotics Laboratory, Department of Computer Science, Stanford University, Palo Alto, CA, 1991. 82 pp.

[Le Pape, 1992] Le Pape, Claude, "Using Constraint Propagation in Blackboard Systems: A Flexible Software Architecture for Reactive and Distributed Systems", *IEEE Computer*, to appear.

[Lynch, 1960] Lynch, Kevin, *The Image of the City* (Cambridge, MA: MIT Press), 1960. 194 pp.

[Lyons and Hendriks, 1992] Lyons, D. M., and A. J. Hendriks, "Reactive Planning", in Schapiro, Stuart C. (Ed.), *The Encyclopedia of Artificial Intelligence* (2nd Edition), 1992, pp. 1171-1181.

[Maes, 1991] Maes, Pattie, "Situated Agents Can have Goals", in Maes, Pattie (Ed.), *Designing Autonomous Agents* (Cambridge, MA: MIT Press), 1991, pp. 49-70.

[Maes and Brooks, 1990] Maes, Pattie, and Rodney A. Brooks, "Learning to Coordinate Behaviours", *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, 1990, pp. 796-802.

[McCarthy, 1959] McCarthy, John, "Programs with Common Sense", *Proceedings of the Symposium on Mechanisation of Thought Processes*, Vol. 1, 1959, pp. 77-84.

[McCarthy and Hayes, 1969] McCarthy, John, and Patrick J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in Meltzer, Bernard, and Donald Mitchie (Eds.), *Machine Intelligence 4* (Edinburgh: Edinburgh University Press), 1969, pp. 463-502.

[McDermott, 1978] McDermott, Drew V., "Planning and Acting", *Cognitive Science*, Vol. 2, 1978, pp. 71-109.

[McDermott, 1981] McDermott, Drew V., "Artificial Intelligence Meets Natural Stupidity", in Haugeland, John (Ed.), *Mind Design* (Cambridge, MA: MIT Press), 1981, pp. 143-160.

[McDermott, 1982] McDermott, Drew V., "A Temporal Logic for Reasoning about Processes and Plans", *Cognitive Science*, Vol. 6, 1982, pp. 101-155.

[McDermott, 1991] McDermott, Drew V., *Robot Planning*, Invited Address, Tenth National Conference on Artificial Intelligence, Anaheim, CA, July, 1991.

[McDermott and Hendler, 1993] McDermott, Drew V., and James Hendler, Call for Papers for a Special Issue of *Artificial Intelligence* on Planning and Scheduling, comp.ai Usenet article, Feb 16, 1993.

[McGee, 1984] McGee, Harold, *On Food and Cooking: The Science and Lore of the Kitchen* (New York : Scribner's), 1984. 684 pp.

[Miller, 1994] Miller, David P., "The Long-Term Effects of Secondary Sensing", *AI Magazine*, Vol. 15, No. 1, Spring, 1994, pp. 52-56.

[Miller et al., 1960] Miller, George A. , Eugene Galanter, and Karl H. Pribram, *Plans and the Structure of Behaviour* (New York: Holt), 1960. 226 pp.

[Minsky, 1981] Minsky, Marvin, "A Framework for Representing Knowledge", in Haugeland, John (Ed.), *Mind Design* (Cambridge, MA: MIT Press), 1981, pp. 95-128.

[Minsky, 1986] Minsky, Marvin, *The Society of Mind* (New York: Simon and Schuster), 1986. 339 pp.

[Minzberg, 1971] Minzberg, Henry, "Managerial Work: Analysis from Observation", *Management Science*, Vol. 18, No. 2, 1971, pp. b97-b110.

[Mintzberg et al., 1976] Mintzberg, Henry, Duru Raisinghani, and André Théorêt, "The Structure of Unstructured Decision Processes", *Administrative Science Quarterly*, Vol. 21, 1976, pp. 246-275.

[Montgomery et al., 1992] Montgomery, Thomas A., Jaeho Lee, David J. Musliner, Edmund H. Durfee, Daniel Damouth, Young-pa So, and the University of Michigan Distributed Intelligent Agents Group (UMDIAG), *MICE Users Guide*, AI Lab, Dept of Electrical Engineering and Computer Science, University of Michigan, January, 1992. 20 pp.

[Neisser, 1976] Neisser, Ulrich, *Cognition and Reality* (San Francisco: W. H. Freeman & Co.), 1976. 230 pp.

[Newell, 1988] Newell, Allen, "The Knowledge Level", in Englemore, Robert (Ed.), *Readings from the AI Magazine, Volumes 1-5, 1980-1985* (Menlo Park, CA: AAAI Press), 1988, pp. 357-377.

[Newell, 1989] Newell, Allen, *IJCAI Excellence in Research Award Address*, Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, 1989.

[Newell and Simon, 1972] Newell, Allen, and Herbert A. Simon, *Human Problem Solving*, (Englewood Cliffs, NY: Prentice-Hall), 1972, 920 pp.

[Newell and Simon, 1976] Newell, Allen, and Herbert A. Simon, "Computer Science as Empirical Inquiry: Symbols and Search", *Communications of the ACM*, Vol. 19, No. 3, March, 1976, pp. 113-126.

[Newell, Shaw, and Simon, 1963] Newell, Allen, J. C. Shaw, and Herbert A. Simon, "Empirical Explorations with the Logic Theory Machine: A Case History in Heuristics", in Feigenbaum, Edward A., and Julian Feldman (Eds.), *Computers and Thought* (New York: McGraw-Hill), 1963, pp. 109-133.

[Nii, 1986] Nii, H. Penny, "Blackboard Systems: The Blackboard Model of Problem-solving and the Evolution of Blackboard Architectures", *AI Magazine*, Vol. 7, No. 2, Summer, 1986, pp. 38-53.

[Nii, 1989] Nii, H. Penny, "Blackboard Systems", in Barr, Avron, Paul R. Cohen, and Edward A. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence*, Vol. IV (Reading, MA: Addison-Wesley), 1989, pp. 1-82.

[Nilsson, 1971] Nilsson, Nils J., *Problem-Solving Methods in Artificial Intelligence* (New York: McGraw-Hill), 1971. 255 pp.

[Norman, 1988] Norman, Donald A., *The Psychology of Everyday Things* (New York: Basic Books), 1988. 257 pp.

[Ortony et al., 1988] Ortony, Andrew, Gerald L. Clore, and Allan Collins, *The Cognitive Structure of Emotions* (Cambridge, UK: Cambridge University Press), 1988. 207 pp.

[Ow et al., 1988] Ow, Peng Si, Stephen S. Smith, and Alfred Thiriez, "Reactive Plan Revision", *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, MN, 1988, pp. 77-82.

[Parker, 1886] Parker, Eliza, *Economical Housekeeping: A complete System of Household Management*, (Toronto: J.S. Robertson & Bros.), 1886. 598 pp.

[Pauker et al., 1976] Pauker, Stephen G., G. Anthony Gorry, Jerome P. Kassirer, and William B. Schwartz, "Towards the Simulation of Clinical Cognition: Taking a Present Illness by Computer", *American Journal of Medicine*, Vol. 60, 1976, pp. 981-996.

[Pearce, 1987] Pearce, John M., *An Introduction to Animal Cognition* (London: Lawrence Erlbaum Associates), 1987. 327 pp.

[Peckham, and Freeland-Graves, 1979] Peckham, Gladys C., and Jeanne H. Freeland-Graves, *Foundations of Food Preparation* (New York: MacMillan), 4th Edition, 1979. 616 pp.

[Pollack and Ringuette, 1990] Pollack, Martha E., and Marc Ringuette, "Introducing the Tileworld: Experimentally evaluating Agent Architectures", in *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, 1990, pp. 183-189.

[Polya, 1957] Polya, George, *How to Solve it: a New Aspect of Mathematical Method* (Princeton, NJ: Princeton University Press), 1957. 253 pp.

[Pylyshyn, 1987] Pylyshyn, Zenon (Ed.), *The Robot's Dilemma: The Frame Problem in Artificial Intelligence* (Norwood, NJ: Ablex), 1987. 156 pp.

[Reilly and Bates, 1992] Reilly, W. Scott, and Joseph Bates, *Building Emotional Agents*, Technical Report CMU-CS-92-143, School of Computer Science, Carnegie-Mellon University, 1992. 13 pp.

[Reisbeck and Schank, 1989] Reisbeck, Christopher K., and Roger C. Schank, *Inside Case-Based Reasoning* (Hillsdale, NJ: Lawrence Erlbaum Associates), 1989. 423 pp.

[Rich, 1983] Rich, Elaine, *Artificial Intelligence* (New York: McGraw-Hill), 1983. 436 pp.

[Ricketts, 1979] Ricketts, M. J., "From Academe to Reality, or How I Learned that Operations Research is Much More than the Application of Quantitative Analysis Techniques", *Infor*, Vol. 17, No. 4, 1979, pp. 394-404.

[Russell and Wefald, 1991] Russell, Stuart J., and Eric Wefald, *Do the Right Thing* (Cambridge, MA: MIT Press), 1991. 186 pp.

[Sacerdoti, 1974] Sacerdoti, Earl D., "Planning in a Hierarchy of Abstraction Spaces", *Artificial Intelligence*, Vol. 5, 1974, pp. 115-135.

[Sacerdoti, 1975] Sacerdoti, Earl D., "The Non-Linear Nature of Plans", *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, 1975, pp. 206-214.

[Sacerdoti, 1988] Sacerdoti, Earl D., "Problem-solving Tactics", in Englemore, Robert (Ed.), *Readings from the AI Magazine, Vols. 1-5, 1980-85* (Menlo Park, CA: AAAI Press), 1988, pp. 521-529.

[Schank, 1982] Schank, Roger C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People* (Cambridge, UK: Cambridge University Press), 1982. 234 pp.

[Schank, 1987] Schank, Roger C., "What is AI, Anyway?", *AI Magazine*, Vol. 8, No. 3, Winter, 1987, pp. 59-65.

[Schank and Abelson, 1977] Schank, Roger C., and Robert Abelson, *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*, (Hillsdale, NJ: Lawrence Erlbaum Associates), 1977. 248 pp.

[Schmidt, 1985] Schmidt, Charles F., "Partial Provisional Planning", in Hobbes, Jerry R. and Robert C. Moore (Eds.), *Formal Theories of the Commonsense World* (Norwood, NJ: Ablex), 1985, pp. 227-268.

[Schoppers, 1987] Schoppers, Marcel J., "Universal Plans for Reactive Robots in Unpredictable Environments", in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, pp. 852-859.

[Schoppers, 1989] Schoppers, Marcel J., "In Defense of Reaction Plans as Caches", *AI Magazine*, Vol. 10, No. 4, Summer, 1989, pp. 51-60.

[Schwartz, 1984] Schwartz, Barry, *Psychology of Learning and Behaviour* (New York: W. W. Norton and Company), 1984. 560 pp.

[Scribner, 1984] Scribner, Sylvia, "Studying Working Intelligence", in Rogoff, Barbara and Jean Lave (Eds.), *Everyday Cognition: Its Development in Social Context* (Cambridge, MA: Harvard University Press), 1984, pp. 9-40.

[Seelig, 1991] Seelig, Tina Lynn, *The Epicurean Laboratory: Exploring the Science of Cooking* (New York: W. H. Freeman & Company), 1991. 163 pp.

[Shortliffe et al., 1979] Shortliffe, Edward H., Bruce G. Buchanan, and Edward A. Feigenbaum, "Knowledge Engineering for Medical Decision Making: A Review of Computer-Based Clinical Decision Aids", *Proceedings of the IEEE*, Vol. 67, 1979, pp. 1207-1224.

[Simon, 1957] Simon, Herbert A., *Models of Man, Social and Rational: Mathematical Essays on Rational Human Behaviour in a Social Setting* (New York: Wiley), 1957. 287 pp.

[Simon, 1969] Simon, Herbert A., *The Sciences of the Artificial* (Cambridge, MA: MIT Press), 1969. 247 pp.

[Simon, 1982] Simon, Herbert A., "Rational Choice and the Structure of the Environment", in Simon, Herbert A. (Ed.), *Models of Bounded Rationality* (Cambridge, MA: MIT Press), Volume 2, 1982, pp 259-268.

[Simon, 1983] Simon, Herbert A., "Search and Reasoning in Problem-solving", *Artificial Intelligence,* Vol. 21, No. 1-2, pp. 7-29.

[Simon, 1993] Simon, Herbert A., "Artificial Intelligence as an Experimental Science", *Invited Address,* Eleventh National Conference on Artificial Intelligence, Washington, DC, July, 1993.

[Smith et al., 1986] Smith, Stephen S., Mark S. Fox, and Peng Si Ow, "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems", *AI Magazine,* Vol 7, No. 4, Fall, 1986, pp. 45-61.

[Stefik, 1981a] Stefik, Mark, "Planning with Constraints", *Artificial Intelligence,* Vol. 16, No. 2, 1981, pp. 111-140.

[Stefik, 1981b] Stefik, Mark, "Planning and Meta-Planning", *Artificial Intelligence,* Vol. 16, No. 2, 1981, pp. 141-170.

[Sternberg, 1982] Sternberg, Robert J. (Ed.), *Handbook of Human Intelligence* (Cambridge, UK: Cambridge University Press), 1982, 1031 pp.

[Sternberg and Salter, 1982] Sternberg, Robert J., and William Salter, "Conceptions of Intelligence", in Sternberg, Robert J. (Ed.), *Handbook of Human Intelligence* (London: Cambridge University Press), 1982, pp. 3-28.

[Suchman, 1987] Suchman, Lucy, *Plans and Situated Actions* (London: Cambridge University Press), 1987. 203 pp.

[Tate, 1975] Tate, Austin, "Interacting Goals and their Use", *Proceedings of the Fourth International Joint Conference on Artificial Intelligence,* Tbilisi, Georgia, USSR, 1975, pp. 215-218.

[Tate, 1976] Tate, Austin, *Project Planning using a Hierarchic Non-Linear Planner,* D.A.I. Research Report No. 25, Department of Artificial Intelligence, University of Edinburgh, 1976. 58 pp.

[Tate and Whitier, 1984] Tate, Austin, and A. M. Whitier, "Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem", in *Proceedings of the First IEEE Conference on AI Applications*, 1984, pp. 410-415.

[Tsang, 1988] Tsang, Edward P.K., "Elements in Temporal Reasoning in Planning", in Ras, Zbigniew W., and Saitta, Lorenza (Eds.), *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems*, Turin, Italy, 1988, pp. 91-100.

[Twain, 1917] Twain, Mark, *What is Man? : and Other Essays* (New York: Harper), 1917. 375 pp.

[Tulving, 1972] Tulving, Endel, "Episodic and Semantic Memory", in Tulving, Endel, and Wayne Donaldson (Eds.), *Organization of Memory* (New York: Academic Press), 1972, pp. 382-403.

[van Melle et al., 1984] van Melle, William, Edward H. Shortliffe, and Bruce G. Buchanan, "EMYCIN: A Knowledge Engineer's Tool for Constructing Rule-Based Expert Systems", in Buchanan, Bruce, and Edward H. Shortliffe (Eds.), *Rule-Based Expert Systems* (New York: Addison-Wesley), 1984, pp. 302-528.

[Vere, 1983] Vere, Steven, "Planning in Time: Windows and Durations for Activities and Goals", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, 1983, pp. 246-267.

[Visser, 1991] Visser, Margaret, *The Rituals of Dinner* (Toronto: Harper Collins), 1991. 432 pp.

[Waterman, 1986] Waterman, Donald A., *A Guide to Expert Systems* (Reading, MA: Addison-Wesley), 1986. 419 pp.

[Webber and Nilsson, 1981] Webber, Bonnie Lynn, and Nils J. Nilsson, *Readings in Artificial Intelligence* (Palo Alto, CA: Tioga), 1981. 547 pp.

[Wilensky, 1983] Wilensky, Robert, *Planning and Understanding: A Computational Approach to Human Reasoning* (Reading, MA: Addison-Wesley), 1983. 168 pp.

[Wilkins, 1980] Wilkins, David E., "Using Patterns and Plans in Chess", *Artificial Intelligence*, Vol. 14, No. 2, 1980, pp. 165-203.

[Wilkins, 1988] Wilkins, David E., *Practical Planning: Extending the Classical AI Planning Paradigm* (San Mateo, CA: Morgan Kaufmann), 1988. 205 pp.

[Winston, 1984] Winston, Patrick *Artificial Intelligence* (Reading, MA: Addison Wesley), Second Edition, 1984. 524 pp.

[Wright, 1990] Wright, Charles E., "Controlling Sequential Motor Activity", in Osherson, Daniel N., Stephen M. Kosslyn, and John M. Hollerbach (Eds.), *Visual Cognition and Action* (Cambridge, MA: MIT Press), 1990, pp. 285-316.

[Yang et al., 1991] Yang, Qiang, Josh Tenenberg, and Steve Woods, *Abtweak: Abstracting a Non-linear, Least Commitment Planner*, Research Report CS-91-65, Department of Computer Science, University of Waterloo, 1991.

[Yeap, 1988] Yeap, Wai K., "Towards a Computational Theory of Cognitive Maps", *Artificial Intelligence*, Vol. 34, 1988, pp. 297-360.