Spatial Reasoning and Integration Techniques for Geophysical and Geological Exploration Data

by

Ping An

This Thesis is Submitted to the Faculty of
Graduate Studies in Partial Fullfillment of
the Requirements for a Degree of

DOCTOR OF PHILOSOPHY

in Geophysics

Geophysics, Department of Geological Sciences
University of Manitoba, Winnipeg, Manitoba

November, 1992

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

ISBN 0-315-81697-X

Canada

SPATIAL REASONING AND INTEGRATION TECHNIQUES FOR GEOPHYSICAL AND GEOLOGICAL EXPLORATION DATA

BY

PING AN

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

DOCTOR OF PHILOSOPHY

Contents

Abstract vi

Acknowledgements vii

List of Figures viii

List of Tables xii

List of Abbreviations xiii

1 Introduction 1

1.1 A General Review of Artificial Intelligence 1

1.2 Knowledge Representation and Inference Mechanisms 3

1.3 Representation and Quantification of Uncertainties 5

1.4 Review of AI Applications in Geosciences 8

1.5 Integrated Resource Exploration 15

2 Test Areas and Data sets 18

2.1 Farley Lake Area 18

2.2 Snow Lake Area 23

3 Set-Theoretic Representation and Integration of Geoscience Data 35

3.1 Geoscience Data Sets as Classical Sets 35
3.2 G-mapping Operations 38
3.3 Integration as Special Types of G-mappings 42
3.4 Integration Using Bayes' Rule 46
3.5 Comment and Discussion 49

4 Fuzzy Set Theory and Integration of Geological and Geophysical Data Sets 50

4.1 Fuzzy Set Theory 52
4.2 Fuzzy Representation 54
4.3 Fuzzy Integration 56
4.4 Test Examples 61
4.4.1 Farley Lake Test Area 62
4.4.2 Snow Lake Test Area 68
4.5 Jackknife Analysis 71
4.5.1 Jackknife Technique 71
4.5.2 Farley Lake Area 73
4.5.3 Snow Lake Area 77
4.6 Error analysis 81
4.6.1 Error Propagation through the Aggregation Processes 81
4.6.2 Interpretation of Relative Information 83
4.7 Results and Discussion 84

5 Evidential Belief Function and Integrating Geological and Geophysical Data Sets 87

5.1 Dempster-Shafer Evidential Belief Function Theory 88
5.2 Dempster's Rule of Combination 91
5.3 Representation of Exploration Data 93

5.3.1 Traditional Probabilistic $X-P$ Mapping 95
5.3.2 Expert $X-P$ Mapping 97
5.4 Integration of Exploration Data Using Dempster's Rule of Combination 98
5.5 Test Examples 103
5.5.1 Farley Lake Test Area 103
5.5.2 Snow Lake Test Area 114
5.6 Error Estimation and Degree of Conflict 119
5.6.1 Error Propagation 119
5.6.2 Degree of Conflict 121
5.7 Discussion and Conclusion 126

6 An Object-oriented and Map-based Expert System 128
6.1 Some Problems in the Application of AI Techniques to Resource Exploration 130
6.2 Integration and Object-Oriented Programming 133
6.3 Object-Oriented Knowledge Representation 135
6.3.1 The Relation Network 135
6.3.2 The Intelligent Objects 139
6.4 Rules in the system 142
6.5 The Inference Engine 143
6.5.1 Search Algorithm 144
6.5.2 Inference Mechanisms 144
6.6 Problems with Dependent Evidence 149
6.7 Some Comments on the Prototype System 151
6.8 Test of the Prototype System 153
6.9 Conclusions on the Prototype Development 158

7 Conclusion 160

Bibliography 163

Appendix A: Objects, Data and Support Files 176

Objects Currently Available in the System 176

The Data and Support Files 176

The Data File Format 176

The Legend Files 176

The Project File 178

Customizing the System 179

The Relation Network File 180

The Rule Base Files 180

Appendix B: Source Code of the Expert System in C^{++} 182

Abstract

An integrated approach is one of the most distinguishing characteristics of exploration for non-renewable resources. The integrated exploration using multiple spatial data sets is mathematically represented and modeled in this thesis. Fuzzy set theory provides an adequate tool to represent imprecise and incomplete geophysical and geological information and to integrate the information for a specific exploration target. Dempster-Shafer evidential belief function provides a more natural basis for representation of geophysical and geological information than the traditional probability approach. Integrated results using belief function consequently provide a more realistic description about a proposition towards the exploration target. The fuzzy set approach and the Dempster-Shafer approach can be used both in areas where there are actual known exploration target occurrences and also in areas where there are as yet no reported target occurrences. An object-oriented knowledge representation structure and corresponding inference mechanisms based on belief function theory are developed for a knowledge-based approach of integrating geophysical and geological data. An object-oriented and map-based prototype expert system is implemented to test this object-oriented knowledge representation structure and its corresponding inference mechanisms for base metal exploration application. The resulting system appears very effective in handling complex integrated exploration problems. The information representation and integration techniques using fuzzy set and belief function approaches, and the prototype expert system are tested with real mineral exploration data from the Farley Lake and Snow Lake areas, Manitoba, Canada. The test results demonstrate considerably improved accuracy and effectiveness in information representation techniques using the fuzzy set and belief function approaches relative to the conventional intuitive approach. The knowledge representation structure and inference mechanisms implemented in the expert system also performed well

and provide a robust theoretical basis for developing a comprehensive expert system for non-renewable resource exploration.

Acknowledgements

I would like to express my greatest appreciation to my advisor Dr. W. M. Moon for his patient supervision and encouragement during this thesis research. I would like to thank Drs. G. F. Bonham-Carter and C. F. Chung at the Geological Survey of Canada for helpful discussions and encouragements. I would like to thank Mr. I. T. Hosain at Energy and Mines, Provincial Government of Manitoba for introducing me to the government open assessment files and his suggestions on choosing the test areas. I would also like to thank Manitoba Mineral Resources Limited for providing some of the test data sets in the Farley Lake area. I am indebted to my wife, *Yi Jiang* for her understanding, continuous support, and patient waiting.

List of Figures

2.1 Location map for the Farley Lake Test Area 19
2.2 Location map for the Snow Lake Test Area 19
2.3 Mineral Occurrence Location Map (Farley Lake). 24
2.4 Bedrock Geology Map (Farley Lake). 24
2.5 Airborne Magnetic Total Field Map (γ) (Farley Lake). 25
2.6 Ground Electromagnetic Map (Farley Lake). 25
2.7 VLF Electromagnetic Contour Map (Station: NNS) (Farley Lake). . . 26
2.8 VLF Electromagnetic Contour Map (Station: NLK) (Farley Lake). . . 26
2.9 IP Chargeability Contour Map (Farley Lake). 27
2.10 Ground Resistivity Contour Map (Farley Lake). 27
2.11 Airborne INPUT Electromagnetic Map (Farley Lake). 28
2.12 Airborne Electromagnetic Map (Farley Lake). 28
2.13 Mineral Occurrence Location Map (Snow Lake). 31
2.14 Ground Electromagnetic Anomaly Map (Snow Lake). 31
2.15 Airborne Magnetic Total Field Map (Snow Lake). 32
2.16 Airborne Electromagnetic Anomaly Map (Snow Lake). 32
2.17 Ground Magnetic Anomaly Map (Snow Lake). 33
2.18 Airborne Magnetic Total Intensity Anomaly Map (Snow Lake). . . . 33
2.19 Geology Map (Snow Lake) 34

4.1 Theoretical Property of γ Aggregation ($\gamma = 0.975$) 59

4.2 Theoretical Curves of Zero Aggregation. 60
4.3 Possibility Map for a Base Metal Deposit using γ-Operator (Farley Lake). 65
4.4 Possibility Map for a Base Metal Deposit using Algebraic Sum (Farley Lake). 66
4.5 Possibility Map for an Iron Formation using γ-Operator (Farley Lake). 66
4.6 Possibility Map for an Iron Formation using Algebraic Sum (Farley Lake). 67
4.7 Possibility Map for a Base Metal Deposit using γ-operator (Snow Lake). 70
4.8 Possibility Map for a Base Metal Deposit using Algebraic Sum (Snow Lake). 70
4.9 Jackknife Estimation of Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake). 74
4.10 Jackknife Estimation of Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake). 74
4.11 Jackknife Variance of Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake). 75
4.12 Jackknife Variance of Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake). 75
4.13 First Order Bias in the Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake). 76
4.14 First Order Bias in the Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake). 76
4.15 Jackknife Estimation of the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake). 78
4.16 Jackknife Estimation of the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake). 78

4.17 Jackknife Variance of the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake). 79
4.18 Jackknife Variance of the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake). 79
4.19 First Order Bias in the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake). 80
4.20 First Order Bias in the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake). 80

5.1 Support ($m(T_I)$) Map for an Iron Formation Deposit (Farley Lake). . 110
5.2 Disbelief ($m(\overline{T}_I)$) Map for an Iron Formation Deposit (Farley Lake). . 110
5.3 Uncertainty ($m(\Theta_I)$) Map for an Iron Formation Deposit (Farley Lake). 111
5.4 Plausibility ($m(T_I)+m(\Theta_I)$) Map for an Iron Formation Deposit (Farley Lake). 111
5.5 Support ($m(T_B)$) Map for a Base Metal Deposit (Farley Lake). 112
5.6 Disbelief ($m(\overline{T}_B)$) Map for a Base Metal Deposit (Farley Lake). . . . 113
5.7 Uncertainty ($m(\Theta_B)$) Map for a Base Metal Deposit (Farley Lake). . 113
5.8 Plausibility ($m(T_B) + m(\Theta_B)$) Map for a Base Metal Deposit (Farley Lake). 114
5.9 Support ($m(T_B)$) Map for a Base Metal Deposit (Snow Lake). 117
5.10 Disbelief ($m(\overline{T}_B)$) Map for a Base Metal Deposit (Snow Lake). 117
5.11 Uncertainty ($m(\Theta_B)$) Map for a Base Metal Deposit (Snow Lake). . . 118
5.12 Plausibility ($m(T_B) + m(\Theta_B)$) Map for a Base Metal Deposit (Snow Lake). 118
5.13 Degree of Conflict (Γ) Map for Base Metal (Farley Lake). 125
5.14 Degree of Conflict (Γ) Map for Iron Formation (Farley Lake). 125
5.15 Degree of Conflict (Γ) Map for Base Metal (Snow Lake). 126

6.1 Block Diagram of the Prototype System 129
6.2 A Relation Network 136
6.3 An ARC Relationship 137
6.4 The OR Relationship 138
6.5 The AND Relationship 139
6.6 Block Diagram of a Terminal Node 140
6.7 Support Map for a Base Metal Deposit (Farley Lake) 154
6.8 Disbelief Map for a Base Metal Deposit (Farley Lake) 155
6.9 Uncertainty Map for a Base Metal Deposit (Farley Lake) 155
6.10 Plausibility Map for a Base Metal Deposit (Farley Lake) 156
6.11 Support Map for a Base Metal Deposit (Snow Lake) 156
6.12 Disbelief Map for a Base Metal Deposit (Snow Lake) 157
6.13 Plausibility Map for a Base Metal Deposit (Snow Lake) 157
6.14 Uncertainty Map for a Base Metal Deposit (Snow Lake) 158

List of Tables

2.1 Description of the Data Sets in the Farley Lake Area. 22
2.2 Description of the Data Sets in the Snow Lake Test Area. 30

4.1 The possibility and probability associated with x 53
4.2 Exploration Data and Membership Functions (Farley Lake). 64
4.3 The Available Exploration D-sets and Membership Functions (Snow Lake). 69

5.1 Propositions and their basic probability measures 100
5.2 Basic Probability Measures for a Base Metal Deposit (Farley Lake). . 106
5.3 Basic Probability Measures for an Iron Formation (Farley Lake). . . . 108
5.4 Basic Probability Measures for a Base Metal Deposit (Snow Lake). . 116

7.1 Available Objects in the System 177

List of Abbreviations

Symbol	Definition and Explanation
$A, B, C, \ldots$	classical sets
x	object of a set
$\varphi(x)$	set specifications
Ψ	an exploration area
M	a geophysical data set
M_a	an aeromagnetic data set
M_b	an aeromagnetic data set in Ψ
X	an exploration set
G	a G-mapping
A_i	a subset of A
B_i	a subset of B
a_i^j	an object of A_i
b_i^j	an object of B_i
$X - F$	mapping a set into a fuzzy set
$X - P$	mapping a set into a probability set
F	a fuzzy set
F_1, F_2	fuzzy subsets
P	a set of probabilities
$P_1, P_2, \ldots$	probability subsets
T	a set of exploration targets or target propositions
T_B	a target subset or proposition for base metal deposits
T_i	a subset of the set T
E	a set of propositions
E_i	a subset of propositions

$=$	equals
$\in$	is a member of
$\rightarrow$	if ... then
$\leftrightarrow$	if and only if
$\exists$	there exists
$\forall$	for all
$\subseteq$	is a subset of
$\subset$	is a proper subset of
$\cup$	union
$\cap$	intersection
$G : A \rightarrow B$	G maps A into B
$A \xrightarrow{G} B$	G maps A into B, the same as $G : A \rightarrow B$
$G_1 \circ G_2$	the composition of the relations G_1 and G_2
$\bigwedge$	for every
$\bigvee$	there exists
D_i	a D-set in X
D_i^j	a subset of D_i
G_i	a subset of the G-mapping
FO	a fuzzy operator
PO	a probability operator
T_t	a set of a specific exploration target T
P_f	favorability
p_j^i	partial favorability
BR	Bayes rule of conditioning
$\mu_F(x)$	membership function of fuzzy set F
$max\{\mu_1, \mu_2\}$	maximum of μ_1 and μ_2
$min\{\mu_1, \mu_2\}$	minimum of μ_1 and μ_2
F_R	fuzzy restriction

$\pi(x)$	a possibility distribution
π	a set of possibility numbers
π_i	a possibility number in π
$P(x)$	a probability distribution
$P(a_i\|T_t)$	probability of a_i given T_t
δ	degree of consistency
p_i	a probability number
θ	unknown value to be estimated
$\widehat{\theta}$	an estimate of θ with n samples
$\widehat{\theta}_{-i}$	an estimate of θ with $n-1$ samples
$\widetilde{\theta}$	jackknife estimation of θ
$E(\widehat{\theta})$	an estimate of $\widehat{\theta}$ with bias
$E(\widetilde{\theta})$	an estimate of $\widetilde{\theta}$
$Var(\widetilde{\theta})$	jackknife variance
γ	a fuzzy operator
$\prod_{i=1}^{n} \mu_i$	product of μ_1 through μ_n
μ_F	a membership number of F
μ_i	a membership number of a fuzzy subset F_i
$\mu_I(x)$	membership function for iron formation
$\mu_B(x)$	membership function for base metal
$\widehat{\mu}$	a membership estimate of μ
ϵ	error term in the estimation of μ
ϵ_i	error from $X-F$ mapping of ith D-set
μ_{-i}	algebraic product with ith term deleted
ϵ_p	error from algebraic product
ε_p	relative error of algebraic product
ϵ_s	error from algebraic sum
ε_s	relative error of algebraic sum
ν	$\prod_1^n(1-\mu_i(x)$

μ_r	membership number when the γ-operator is used
ϵ_r	error associated with the γ-operator
ε_r	relative error associated with the γ-operator
Θ	a frame of discernment
2^Θ	a set of all subsets of Θ
$\emptyset$	an empty set
H, $H_1, H_2, \ldots$	subsets of Θ
$S, S_1, S_2, \ldots$	subsets of Θ
$Bel(H)$	a belief function
$m(H)$	probability measure of H
$Pl(H)$	a plausibility function for H
DRC	Dempster's rule of combination
k	probability committed to $\emptyset$
Q	an attribute in a D-set
N	total squares in the Ψ
N_Q	number of squares with attribute Q
N_{QT}	number of squares with Q and T_i
T_I	target proposition for iron formation deposits
$\overline{T}_I$	negation of T_I
ϵ_{Hi}	error term of $m_1(H_i)$
ϵ_{Sj}	error term of $m_2(S_j)$
ϵ_H	error estimation of $m(H)$
K	re-normalizition constant
Γ	degree of conflict
D-S	Dempster-Shafer

Chapter 1

Introduction

In this chapter, a brief review of artificial intelligence (AI) techniques is presented. It includes their definitions, developments, associated theories to quantify uncertainties, knowledge representation techniques and inference mechanisms. This is followed by a brief review of geoscience applications of the AI techniques. Finally, integrated resources exploration, one of the most important geological application fields of the AI techniques, is introduced.

1.1 A General Review of Artificial Intelligence

Artificial intelligence (Shapiro et al., 1987; Rich, 1983; Winston, 1984) is a domain of research, application, and instruction that includes programming of computers to perform in ways that, if observed by human beings, would be regarded as intelligent. It includes cognitive tasks, at which people, at present, are better. Examples include commonsense tasks, such as understanding English, and recognition of remotely sensed images.

Study of expert systems is an important branch of AI. Expert systems are sophisticated computer programs that manage human knowledge to solve problems

efficiently and effectively in specific problem areas. Some of the tasks of an expert system include locating mineral deposits, diagnosing diseases.

Both the scientific and engineering application disciplines of AI are based on the hypothesis of physical symbol systems. The processes which are required to produce intelligent action can be simulated with a collection of physical symbols and a set of mechanisms that produce a series of structures built with these symbols. In an AI program symbolic structures are used to represent both general knowledge about a problem domain and specific knowledge about the solution to the current problem.

The science of AI began and some of the first AI programs, which played chess and proved theorems, were written in 1950s (Rich, 1987). Starting in the early sixties, several efforts emerged to build programs to do a variety of tasks, primarily ones we now call commonsense tasks. Among these efforts were GPS, the General Problem Solver, which was built by Allen Newell, J. C. Shaw, and Herbert Simon at Carnegie-Mellon University (Ernst and Newell, 1969); STRIPS, a general problem solver built at the Stanford Research Institute (Fikes and Nilsson, 1971); chess and checker programs (Rich, 1983). None of these was particularly successful if success is meant to indicate the construction of programs that could solve significant problems. But out of them emerged two important ideas. The first is understanding the role of search in problem solving; and the second is appreciation of the key role being played by knowledge in controlling the search. The second idea arose directly out of failures of the early systems. The main reason for the failures is now understood to be inadequate knowledge. By the mid-1970s, it was recognized that each program needed to possess knowledge on their domains. For example, MYCIN was designed and implemented by E. H. Shortliffe and others to diagnose and recommend treatment of infectious blood diseases (Gordon and Shortliffe, 1985). The MYCIN program consisted of both a code portion and a knowledge base that was used by the code. The knowledge base contained a set of condition-action rules that described the diagnostic knowledge that

a human physician might possess. Although MYCIN was never used outside of the research laboratory, it demonstrated that tasks that previously had been done only by highly trained human experts might be accomplished by machines. This was the first step toward the current technology of expert systems development, in which programs that solve restricted problems in a wide array of scientific, engineering, commercial, and military areas are being built. PROSPECTOR (Duda, 1979), the first expert system in the geosciences, was designed and implemented during this period at Stanford Research Institute (SRI). By the end of 1970s AI was a thriving research area in a small number of academic and industrial laboratories, but it had little impact outside of those laboratories. In the early eighties, the first generation of expert systems that performed complex tasks in a cost-effective fashion began to appear. Since then, there has been an explosive growth of interest in AI, both in research and applications. In research, people attempt to find long-term answers to many unsolved problems. In applications, many developmental efforts are made in an attempt to apply the discovered techniques to the problems for which those techniques can provide adequate near-term solutions. In 1990 alone, over 1000 systems have been documented to be in production use and the actual number is estimated to be over 3000 systems (Fox, 1990). Almost all these systems are in the areas of engineering and manufacturing.

1.2 Knowledge Representation and Inference Mechanisms

The techniques that AI applies to solve problems involve knowledge representation methods and inference methods for handling the relevant knowledge, and search-based problem-solving methods for exploiting that knowledge. Although the tasks with which AI is most concerned appear to form a very heterogeneous set, they are

related through their common reliance on techniques for manipulating knowledge and conducting search.

There are hundreds of techniques for representing knowledge in AI systems if details are considered. But if details are ignored, those techniques fall into three categories: logical formulas, slot-and-filter structures (Rich, 1983), and production rules. In some cases, such as proving theorems, knowledge can be represented as logical formulas, then all of the formal power of a logical theorem-prover can be applied to the knowledge to generate new knowledge. Slot-and-filler structure knowledge representation uses a network of nodes connected by relations and organized into a hierarchy. Each node represents an object that may be described by attributes and values associated with the object. This method provides a natural and efficient way to represent the knowledge structure associated with a multidisciplinary problem. Rule-based knowledge representation concentrates on the use of -IF *condition* THEN *action*- statements. When the current problem situation satisfies or matches the IF part of a rule, the action specified by the THEN part of the rule is performed. This action may affect the outside world, may direct program control, or may instruct the system to reach a conclusion. The search procedure can be applied to the knowledge base to find a path from the initial state to the goal state.

An AND/OR graph (Pearl, 1987) is an explicit representation of the relationship between all situations and options that may be encountered in the solution of decomposeable problems. The nodes in an AND/OR graph represent subproblems to be solved or subgoals to be achieved, with the top node representing the specification of the overall problem. The nodes are connected by two types of links: OR links and AND links. The OR links represent alternative options of handling the problem node from which they emanate. The problem is solved if any one of the subproblems is solved. The AND links connect a parent node to individual problems of which it is composed. The parent problem is solved only when all its subproblems are solved.

A terminal node in an AND/OR graph represents either a primitive problem whose solution is readily available or a subproblem that cannot be decomposed any further. The basic idea of the AND/OR graph is very useful to knowledge representation for information integration tasks. It can be used to explicitly represent the relations which are relevant to the overall exploration target between different data and concepts. Actually, this idea, combined with frame and rule-based techniques, is used in the expert system which is developed in this thesis research for integrating geoscience information for resource exploration. It will be described in detail in Chapter 6.

1.3 Representation and Quantification of Uncertainties

In the real world, one frequently faces tasks that defy absolute or categorical analysis because of their complexity, lack of sufficient knowledge, and incomplete and imprecise evidence (data). Typical examples of this type include prediction of a mineral deposit and diagnosis of a disease. It is however very important in the real world to reach a decision based on imprecise evidence and using incomplete knowledge. In such situations, proper uncertainty representation and inexact but careful reasoning are crucially important.

Three major mathematical theories which can be applied to expert systems for management of uncertainties are Bayesian probability theory, Dempster-Shafer evidential belief function theory, and fuzzy set theory. During the mid-1980s, there were an increasing number of discussions about these theories both in the field of AI and in statistics. The topics in the field of AI are mainly focused on how to apply these theories to manage uncertainties in specific knowledge structures and corresponding inference mechanisms, such as Gordon and Shortliffe (1985), Lee and Shin (1987), Lu and Stephanou (9184), Dubois and Prade (1985), and Zedeh (1986).

In the field of statistics, there have recently appeared an increasing number of discussions on how a certain theory would be suitable to handle the uncertainties in expert systems. Lindley (1987), a leading proponent and developer of the use of the subjective probability and Bayesian theory for representation of uncertainties writes: "Our thesis is simply stated: the only satisfactory description of uncertainty is probability. By this is meant that every uncertainty statement must be in the form of a probability; that several uncertainties must be combined using the rules of probability; and that the calculus of probabilities is adequate to handle all situations involving uncertainty. In particular, alternative descriptions are unnecessary. These include ... possibility statements in fuzzy logic, ... and belief functions." Shafer (1987), the major proponent for the use of the belief function theory to represent uncertainties, writes: "After developing a constructive understanding of the Bayesian theory, I introduce another constructive theory, the theory of belief functions. I argue that both theories should be thought of as languages for expressing probability judgements and constructing probability arguments." Spiegelhalter (1987), a leading statistician working on the use of expert systems in medicine, writes: "The development of expert systems in medicine has generally been accompanied by a rejection of formal probabilistic methods for handling uncertainty. We argue that a coherent probabilistic approach can, if carefully applied, meet many of the practical demands being made." In commenting on the papers of the above three authors, Dempster and Kong (1987) of Harvard University wrote: "In our view, the belief function system is very close to Bayes, and indeed includes Bayesian models as a special case, ... ". Watson (1987) of Cambridge University, who has himself done important works related to expert systems, states: "Probability theory has a strong intellectual support and in principle there is no reason why one should not be satisfied with this theory. Its use does, however, lead to enormous problems of complexity, and as a matter of practice it is necessary to seek for approximations. Fuzzy set theory can be viewed as a heuristic for

handling those situations where imprecise inputs and imprecise inference are required without the need to resort to the greater complexity of probability theory. Belief function theory can be thought of as a way of representing inferences from evidence within the probabilistic framework."

The statements given by the above authors clearly provide contrasting insights into these theories. At this point, after considerable efforts were made to apply these theories to geological problems, particularly to mineral resource exploration problems, I find myself being in more agreement with Watson's view. As demonstrated in this thesis, both fuzzy set theory and belief function theory work well for integration of geoscience data for mineral resource exploration. They can be conveniently applied to both less-explored areas and well-explored areas. Bayesian probability theory can be a good choice for well-surveyed regions with a training area. In the development of the expert system in this thesis research, belief function theory is utilized to represent and process uncertainties. The effects of incomplete spatial coverage of certain data sets can be better represented in a map-based system and mathematical and computational complexity can be avoided if the knowledge structures are designed in a way in which the problem with dependent information can be resolved.

During the early to mid-1980s, statisticians developed methodologies, both in the Bayesian probability and in the belief function frameworks, for probability fusion and propagation using structured knowledge representation models (Pearl, 1985; Shafer and Logan, 1985; Shenoy and Shafer, 1986). Up to now, no report has appeared on successful application of these models to real world problems. I have tried to apply those knowledge representation models. I found that either the models are too simple or not applicable due to the mathematical requirements and complexities which are often impossible to resolve for incomplete and imprecise input data. Another important issue in uncertainty management concerns proper interpretation of the output probability. When a system asks users to supply judgements about certain

evidence, different values can be given to the same evidence by different users, and hence different results can be achieved. Another difficulty in interpreting the certainty output is more intrinsic because a theoretically correct and accurate implementation of a real world problem is still almost impossible. Assumptions have to be made to simplify the problems and uncertainty processing. Many systems being utilized today employ an ad hoc way of handling uncertainties. Further studies are required for representation and quantification of uncertainty in the field of AI and particularly in the geoscience application of AI.

1.4 Review of AI Applications in Geosciences

The first serious application of AI to geoscience began with the construction of an expert system named PROSPECTOR (Duda et al, 1979; Duda, 1980; Duda and Reboh, 1984) after which many expert systems have appeared in geosciences, such as DIPMETER ADVISOR, LITHO, MUD (Waterman, 1986), muPROSPECTOR (McCommon, 1986) and XEOD (Shultz et al, 1988). The AI applications in geoscience are mostly implementation of expert systems. Among the many systems, PROSPECTOR is probably the most widely-quoted one although it was built about fifteen years ago. It was designed at the SRI for decision making processes specifically in mineral exploration and has the distinction of being the first computer expert system built to assist geologists in evaluating mineral prospects.

The work on PROSPECTOR started at the SRI in 1974 and continued until 1983, which included construction of the system, field testing, and evaluation. At the moment PROSPECTOR is the most comprehensive expert system in geosciences. The 12 prospect-scale models in PROSPECTOR contain over 1000 rules and more than 1500 spaces. The 23 regional-scale models in it include over 1300 rules and more than 1600 spaces. Over the course of developing the system, nine different mineral exploration

experts contributed their skills and expertise, working with several knowledge engineers and computer programmers. It was implemented in INTERLISP and it took more than 30 person-years of effort to produce the system. Recently, PROSPECTOR is expanded to be able to combine different map data (McCommon, 1988). The central idea behind PROSPECTOR is to encode the ore-deposit models developed by expert economic geologists in a form that allows the models to be systematically interpreted by a computer program for a variety of purposes. The PROSPECTOR system used a combination of rule-based and semantic net to represent knowledge. The muPROSPECTOR and muPETROL, which were built three years after the PROSPECTOR, adopted the PROSPECTOR's approaches to represent knowledge and applied a simplified inference procedure of the PROSPECTOR.

The PROSPECTOR was originally designed for the following main applications (Duda, 1980):

–Prospect evaluation,

–Regional resource evaluation and

–Drilling site selection.

For prospect evaluation, it can be used by a field geologist in an interactive mode. A field geologist would view the program as a computer-based consultant, providing the system with information about the particular site to determine how well the provided information matches the various models, what additional data would be most worth acquiring to reach firmer conclusions, and why a particular piece of information should be sought and particular conclusions could be reached. In this type of application, the PROSPECTOR both improves the evaluation of available data and reduces the chance of overlooking any particular data segments.

For regional resource evaluation, the PROSPECTOR would probably be used in an off-line mode to process tabulated data about a large number of geological districts.

Here the major purpose would be to evaluate in a systematic and unified manner the degree to which each district is favorable for the development of a particular class of ore deposits.

For drilling site selection, the PROSPECTOR can only be used in a graphic-input mode, in which various kinds of geologic maps are digitized and processed in accordance with specific drilling-site-selection models. Here it assumes that the presence of a specific type of deposit has been already determined and a drilling program is being planned, and that the main task would be to produce an output map that uses an appropriate model to combine the input data to determine the most favorable drilling sites. Such an exploitation of expert knowledge is expected to lead to significant savings in time and the cost of exploratory drilling.

The rules in the PROSPECTOR form a large inference net, which includes connections between evidence and hypotheses and hence all the possible chains that can be generated from the rules. When a new certainty factor of an evidence is given by the user, the program will modify the certainty factor of the subsequent propositional assertions. The part of the system that actually propagates the probabilities forward through the inference net is termed the PROSPECTOR's inference engine.

During the development of the PROSPECTOR, great efforts were made on uncertainty processing. The PROSPECTOR uses Bayesian probability theory to guide the updating (propagation) of the probabilities when new evidence is acquired or added. That is, when a user provides new information, the inference engine updates appropriate probabilities from prior to posterior values. However, a theoretically correct implementation of the Bayesian approach would require knowledge of the joint probability function for all of the events in the model. It is difficult to estimate the joint probability functions for even a small subset of events, and impossible to do so for a relatively large model. Certain assumptions have to be made to update the probabilities in the real problem implementation. As indicated by Konolige (1979),

one of the major contributors to the PROSPECTOR, "some of the assumptions lead to behavior that might eventually be unacceptable", when certain assumptions are erroneous.

The expert systems in geoscience have been built for different purposes, often using different computer languages. If one investigates these systems more closely, one finds that they often utilize different knowledge representation methods, and consequently, different inference methods. In the following, examples of expert systems specifically designed for geoscience applications are briefly reviewed.

LITHO

assists geologists in interpreting well log data. The data include measurement curves of density, resistivity, acoustic wave transmission velocity, and radioactive emission. The system uses the log data in addition to the region's geological environmental knowledge to characterize geological formations encountered in the well. This characterization includes porosity, permeability, composition, texture, and type of layering. The LITHO uses a separate pattern recognition program to extract features directly from the well log data. The knowledge is represented as rules and is accessed through a backward chaining. The LITHO is implemented in EMYCIN and was developed by Schlumberger Co. (Bonnet and Daham, 1983).

muPROSPECTOR

is a mineral consultant system. It assists geologists in evaluating mineral potential, especially of large areas in which the available data are of a reconnaissance nature. Such areas may contain undiscovered mineral deposits of a wide variety of deposit types. In these situations, geologists can be at loss to consider all of the possible deposit types that may be suggested by the data. External to the program is a knowledge base that consists of mineral deposit models. It uses a combination of rule-based and semantic net formalism to represent its knowledge. It estimates the

likelihood of occurrence of the deposit types represented in the knowledge base. The system was implemented in muLISP on an IBM PC or a compatible machine. It was developed by McCommon (1986) after the PROSPECTOR system developed at the SRI International.

muPETROL

is an expert system to aid petroleum geologists in classifying the world's sedimentary basins. The basin classification is based on nine sedimentary basin models using H. D. Klemme's recognition criteria (Klemme, 1980). The knowledge of the system is represented in rule-based models. It uses a forward chaining inference method to evaluate the likelihood of one or more hydrocarbon occurrences in a basin on the basis of the geological characteristics that categorize the basin. The muPETROL is a microcomputer-based petroleum geology expert system, constructed on an IBM PC or a compatible machine in muLISP (Miller, 1987).

MUD

helps engineers to maintain optimal drilling fluid properties. It performs the task by diagnosing causes of problems with drilling fluids and by suggesting appropriate treatments. Possible causes to be diagnosed include parameters such as contaminants, temperature, pressure, and inadequate use of additives. The MUD contains knowledge from the domain experts about drilling fluids and the diagnosis of drilling problems. It is a forward chaining rule-based system and uses certainty factors to represent the subjective determination of experts. In addition, it can provide explanations about its recommanded treatment plans. The MUD is implemented in OPS5 and it was developed at the Carnegie-Mellon University in cooperation with NL Baroid (Kahn and McDermott, 1984).

XEOD

is a microcomputer-based expert system for interpretation of environments of

siliciclastic deposition. The XEOD incorporates rules and semantic nets to represent knowledge. A certainty judgement from 0 to 1 is supplied as input to indicate the user's observational confidence. The calculation of certainty sums is based on an ad hoc method with no statistical basis. The system is implemented in PASCAL (Shultz et al., 1988).

In addition to the systems listed above, Visher and Sutterlin (1990) developed De-Xpert, an expert system to help geoscientists identifying depositional systems. Fowler (1987) developed the SRPS, a knowledge-based system for predicting source rock potentials. Penas et al. (1986) developed an expert system for determining vibroseis field parameters. Miller (1990) reported an object-oriented expert system for sedimentary basin analysis under development at the U. S. Geological Survey. Zhang and Simaan (1991) reported a rule-based system for interpretation of seismic sections based on texture. Lu and Cheng (1991) reported the implementation of an expert system for structural style identification. There are also many other expert systems reported by authors such as Kuo and Startzman, 1987; Conrad and Beightol, 1988; Fang et. al. 1986; Walker, 1988; Coppens, 1991; Veezhinathan, 1991; Fang et al., 1991.

Another equally important AI application field in geoscience is geological remote sensing. Recently, efforts have been made to apply a knowledge-based approach to automate interpretation of remote sensing data. Argialas (1990) reviewed computer techniques, including the expert system approach for remote sensing image interpretation. Egenhofer and Frank (1990) reported their efforts in combining AI and database techniques for a geographical information system (GIS). Campbell and Cromp (1990) identified an urgent need for an intelligent information fusion system for managing enormous volume and complexity of the remote sensing data. Wang (1989) developed an expert system for remote sensing image analysis. Fuzzy set theory was applied in the system to handle uncertainty which originates from the intrinsically imprecise

geographical information derived from remote sensing imagery. In Wang's system, the knowledge is represented as production rules accompanied by certainty factors which denote the certainty level of the rules. Goodenough et al. (1989) applied a knowledge-based technique to analyze the environmental change from remote sensing data. Schenk and Zilberstein (1990) recently examined the feasibility and applicability of building a knowledge-based system to control the interpretation of linear features in digitized topographic maps. Moller-Jensen (1990) developed an expert system for classification of an urban area using texture and context information in Landsat-TM imagery. Hadipriono et al. (1990) developed a knowledge-based system for analysis of drainage patterns using an expert system shell named PC Plus. These list some of the recent efforts and important application potentials.

Most of the expert system developments in the remote sensing discipline are incorporated with GISs that can provide convenient database management, display and basic graphical processing capabilities. Although problems in the remote sensing discipline seem quite different from those in resource exploration, the imprecise nature of the data, knowledge representation, and approximate reasoning processes are very similar to those in the resource exploration. In addition, remotely sensed imagery is often one of the important data sets in resource evaluation or exploration.

Expert system research and subsequent developments after the PROSPECTOR are mostly focused on application of the knowledge extraction, knowledge encoding, and inference techniques, most of which are already well established in the main stream of AI. In most cases, expert system shells are used. An expert system shell usually provides users with built-in knowledge representation structure, inference engine, and methods for managing uncertainties. It provides a convenient tool for quick experiments. But the built-in knowledge representation structure and method for uncertainty management may not be suitable for specific problems. Although these research and developments show important potential values of applying AI techniques

to resource exploration, most of them, however, show little interest and effort, and consequently little progress in representation of imprecise and incomplete knowledge, inference mechanisms, and uncertainty processing for resource exploration. Some systems provide probabilities or certainty factors accompanying their output, but they are inadequate with very simplified assumptions. These are very important issues directly related to effective application of AI techniques to geoscience problems and related to the quality of expert systems.

Another important characteristic of recent expert system research and development is that they are often constrained to specialized domains, such as well-log interpretation and identifying depositional environments. One of the reasons for this characteristic is probably the common belief that AI performs better in a small and well-defined domain. Exploration tasks in resource industry are, however, multidisciplinary. Integrated resource exploration which combines all types of geoscience data is in general much more complex and difficult than the traditional AI applications.

1.5 Integrated Resource Exploration

An integrated approach is one of the distinguishing characteristics of today's nonrenewable resource exploration. With the introduction of new efficient data acquisition techniques and subsequent rapid accumulation of exploration data, the number of data sets available is often large in many exploration areas. The tasks of integrated interpretation for resource exploration are also becoming more complicated with rapidly increasing volume of data. Use of the conventional intuitive approach is becoming less efficient with increasing volume, size, and complexity of data. An approach one can take to resolve this problem of inefficiency is the application of a commonly available GIS. But there is no one GIS package that can satisfy all requirements and be used accurately and efficiently to integrate information from various sensors for resource

exploration and evaluation. A number of statistical and mathematical approaches have recently been developed to complement shortcomings of the current generation of GISs. The Bayesian, regression, and weights of evidence models (Fabbri, 1984; Singer and Kouda, 1988; Agterberg et al., 1990; Agterberg, 1989; Bonham-Carter et al., 1988, 1989) derive prediction maps based on known mineral occurrences in the exploration area. The decision-tree approach (Reddy and Bonham-Carter, 1991) used a commercial software called "Knowledgeseeker" to extract information from known mineral deposits and apply the knowledge to predict new mineral potentials. These new approaches provide us with more objectively generated mineral potential prediction maps based on given data. Problems still exist in precise representation of information and further research is needed because most of the exploration areas are under-explored and there are too few known mineral occurrences. Evidential belief function (Moon, 1990) and fuzzy set (An et al., 1991) approaches can be used in these cases. The evidential belief function approach has the potential of combining known information on mineral occurrences in a chosen exploration area with known mineral occurrence information in geologically similar areas. Difficulties in representing exploration data in an evidential space or in a fuzzy space due to unfamiliarity with belief function and/or fuzzy logic and possible unfamiliarity with all available data can impede their widespread application. An automated expert system can be a solution which provides a useful tool to overcome this difficulty (An et al., 1992).

The tasks of integrated exploration, in the AI perspective, are much more difficult than many well-defined single problems. First, it needs knowledge about specific disciplines from which there is a data set or data sets to be integrated. These disciplines may include geology, geophysics, remote sensing, geochemistry, etc. Often the human experts do not have expertise in all these fields. The second reason is that the data sets from different disciplines often have different data formats. The data are often symbolic in geology and numeric in geophysics. This causes difficulties in knowl-

edge representation and consequent inference processing. These problems naturally suggest combination of several different knowledge representation techniques.

Most of the data sets from different geoscience disciplines are imprecise, incomplete, and in different formats. The knowledge for resource exploration is also not absolute. For example, an explorationist can never simply answer "yes" or "no" for a specific exploration target before it has been observed directly or drilled, even when the survey data available are of a very high quality. Mathematical and/or statistical representation and processing of uncertainties from imprecise and incomplete data, and incomplete knowledge are important issues for AI applications to resource exploration. This research focus on mathematical representation and integration of geological and geophysical data for mineral exploration. This thesis will first briefly review the test areas and test data sets (chapter 2), and then will describe set-theoretic representation of integrated resource exploration (chapter 3), application of fuzzy set theory (chapter 4) and belief function theory (chapter 5) for integration of different geoscience data, and finally, will describe the experiment of knowledge-based approach (chapter 6).

Chapter 2

Test Areas and Data sets

Two test areas were chosen for implementation of the new approaches of this thesis and development of a prototype expert system in this thesis research. One of the test areas is located in the Farley Lake area (Figure 2.1) and the other is located in the Snow Lake area (Figure 2.2), both of which are in northern Manitoba, Canada.

2.1 Farley Lake Area

The Farley Lake test area is located in the northeast part of the Lynn Lake greenstone belt. This geological belt comprises moderately to highly metamorphosed volcanic, sedimentary and plutonic rocks, extending 130 km from Laurie Lake near the Manitoba-Saskatchewan border in the west to Magrath Lake in the east (Syme, 1985). Rocks in the belt represent early Proterozoic deposition, intrusion and metamorphism (Clark, 1980). The volcanic and volcaniclastic rocks belong to the Wasekwan Group (Campbell, 1969) and were intruded by subvolcanic plugs, and subsequently folded, faulted and intruded by larger mafic and felsic plutons. The Wasekwan Group rocks are uncomformably overlain by sandstone and conglomerate of the Sickle Group (Campbell, 1969). Regional metamorphism and further deformation and plutonism

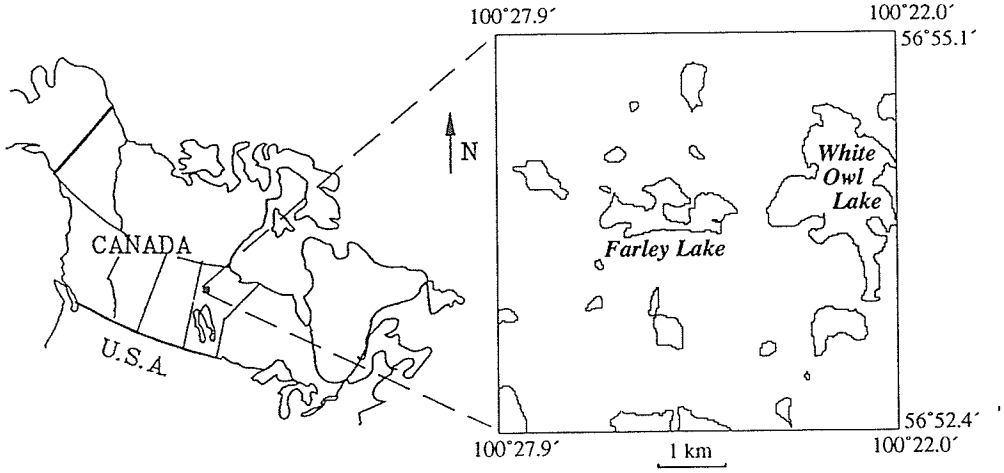

Figure 2.1: Location map for the Farley Lake Test Area

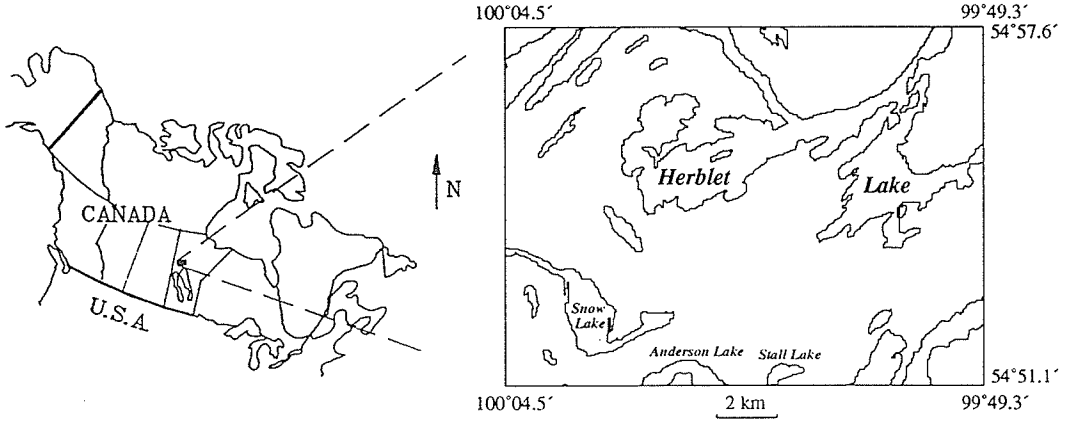

Figure 2.2: Location map for the Snow Lake Test Area

took place after the deposition of the Sickle Group (Syme, 1985). The Lynn Lake greenstone belt is divided into two structural sub-belts: a northern sub-belt between Motriuk Lake and Eagle Lake and a southern sub-belt between Fox Mine and Hughes Lake (Syme, 1985). The two sub-belts are separated by large granitic intrusions.

Magnetite and/or pyrite-pyrrhotite-bearing iron formation occurs within the Wasekwan Group of the northern belt and in the area southwest of Nail Lake in the southern belt. Outcrops occur south of Stear Lake and west of Boiley Lake (Gilbert et al., 1980). The iron formation are commonly associated with electromagnetic anomalies and strong aeromagnetic anomalies.

Nickel-copper (Ni-Cu) and copper-zinc (Cu-Zn) deposits were found in the Lynn Lake greenstone belt. The Ni-Cu deposits occur within mafic to ultramafic igneous plutons of the Wasekwan Group (Pinsent, 1980). The Cu-Zn deposits occur in felsic to mafic metavolcanic rocks of the Wasekwan Group (Gilbert et al., 1980; Syme, 1985).

The Farley Lake test area is located in the northern sub-belt. It covers an area of 6x6 km^2. The bedrock is mostly overlain by surficial deposits which include glacial till, glaciolacustrine silts and clays. Most of the area is covered with fen and bog veneer (Nielson and Graham, 1985) and the geological map is mainly based on very limited outcrops, drill holes and geophysical data.

The mineral deposits discovered in this area are sulphide facies iron formation and gold deposits (Figure 2.3). The iron formation was discovered by Hudson Bay Exploration and Development Limited by drilling ground EM anomalies west of the test area (Chevillard and Genaile, 1970). The gold deposits were discovered in the west central part of the test area. Occurrence 2 (Bamburak, 1990) and occurrence 3 (by personal communication) were discovered by Manitoba Mineral Resources Limited.

Nine maps are available in this test area. The descriptions of these maps are

Map Name	Description	Source of Data
Bedrock Geology Map (Figure 2.4)	Scale 1:25,000.	Manitoba Mineral Resources Limited Project 654, 1984.
Airborne Magnetic Total Field Map (Figure 2.5)	Scale 1:50,000.	Geological Survey of Canada, Open File Report 1047, 1984.
Ground Electromagnetic Map (Figure 2.6)	Horizontal Loop, Frequency=2400 Hz, Scale: 1:4800.	Chevillard and Genaile 1970.
VLF Electromagnetic Contour Map STN: NNS (Figure 2.7)	Horizontal Loop, Station: NNS at Annapolis, USA Frequency=21.4 kHz, Scale 1:2400.	Manitoba Mineral Resources Limited, Project 654, 1986.
VLF Electromagnetic Contour Map STN: NLK (Figure 2.8)	Horizontal Loop, Station: NLK at Seattle, USA Frequency=24.8 kHz, Scale 1:2400.	Manitoba Mineral Resources Limited, Project 654, 1986.
IP Chargeability Contour Map (Figure 2.9)	Time Domain IP, Pole-Dipole, Scale 1:2400.	Manitoba Mineral Resources Limited, Project 654, 1986.
Ground Resistivity Contour Map (Figure 2.10)	Pole-Dipole Scale 1:2400.	Manitoba Mineral Resources Limited, Project 654, 1986.

Table 2.1: Description of the Data Sets in the Farley Lake Area.

Map Name	Description	Source of Data
Airborne INPUT Electromagnetic Anomaly Map (Figure 2.11)	6 channels, Mean Flight Spacing 200 m, Scale 1:20,000.	Questor Surveys Limited, 1969.
Airborne Electromagnetic Anomaly Map (Figure 2.12)	Scale 1:50,000.	INCO, 1956.

Table 2.1: Description of the Data Sets in the Farley Lake Area.

listed in Table 2.1. The anomalies of the Airborne INPUT EM Map were originally represented as number of observed anomaly channels. A good conductor tends to cause higher number of anomaly channels. The anomalies of the Airborne EM Map were represented by their relative anomaly amplitude, decreasing with a sequence of characters A, B, C, D, E. Only lithological types are digitized from the Bedrock Geology Map.

Nine data sets were obtained by digitizing and processing these maps into digital raster maps in the computer. The data matrix size of the digitized raster maps is 256x256. The pixel size is about 24x24 m^2. Their grey level plots are shown in Figures 2.4-2.12.

Only the Bedrock Geology Map (Figure 2.4), Airborne Magnetic Total Field Map (Figure 2.5) and Airborne INPUT Electromagnetic Anomaly Map (Figure 2.11) cover the whole test area. The Airborne Electromagnetic Anomaly Map (Figure 2.12) covers more than half of the test area. The Ground Electromagnetic Map (Figure 2.6), IP Chargeability Contour Map (Figure 2.9), Ground Resistivity Contour Map (Figure 2.10), and the two VLF Electromagnetic Contour Maps (Figures 2.7 and 2.8) only cover a very small portion of the test area. Clearly, the spatial data coverage is very

unbalanced.

2.2 Snow Lake Area

The Snow Lake test area is located in the eastern part of the Flin Flon greenstone belt. This belt comprises a metamorphosed assemblage of early Proterozoic subaqueous and subordinate subaerial volcanic and associated sedimentary rocks, an overlying sequence of terrestrial sedimentary rocks, and a complex array of intrusive rocks (Bailes and Syme, 1989). Metamorphic grade in the Snow Lake area varies from greenschist metamorphism in the volcanic terrane in the south to amphibolite metamorphism in the gneiss terrane to the north (Froese and Moore, 1980; Ferreira and Fedikow, 1991; Bailes and Galley; 1991).

The Flin Flon greenstone belt is well-known for its base metal production from volcanogenic massive Cu-Zn and Zn-Cu sulphide deposits. These deposits mainly occur in felsic to mafic volcanic rocks and usually are associated with electromagnetic anomalies (Fedikow et al., 1989).

The Snow Lake test area is located in the east part of the Flin Flon greenstone belt and covers an area of 16x12 km^2. Cu-Zn deposits occur near the Anderson Lake and Stall Lake (Figure 2.13) (Bailes and Galley, 1991; Ferreira and Fedikow, 1991). There are also gold deposits near the Snow Lake (Fedikow et al., 1989) (Figure 2.13).

Six geological and geophysical data sets are available in this test area. Brief descriptions and their origins are listed in Table 2.2. They were digitized and processed into raster maps in the computer. Their corresponding grey level plots are show in Figures 2.14- 2.19. The raster matrix size is 400x300 and the pixel size is 40x40 m^2.

The Ground EM Anomaly Map (Figure 2.14 was compiled from more than ten different ground EM surveyes in the area by Hosain (1988). The anomalies were

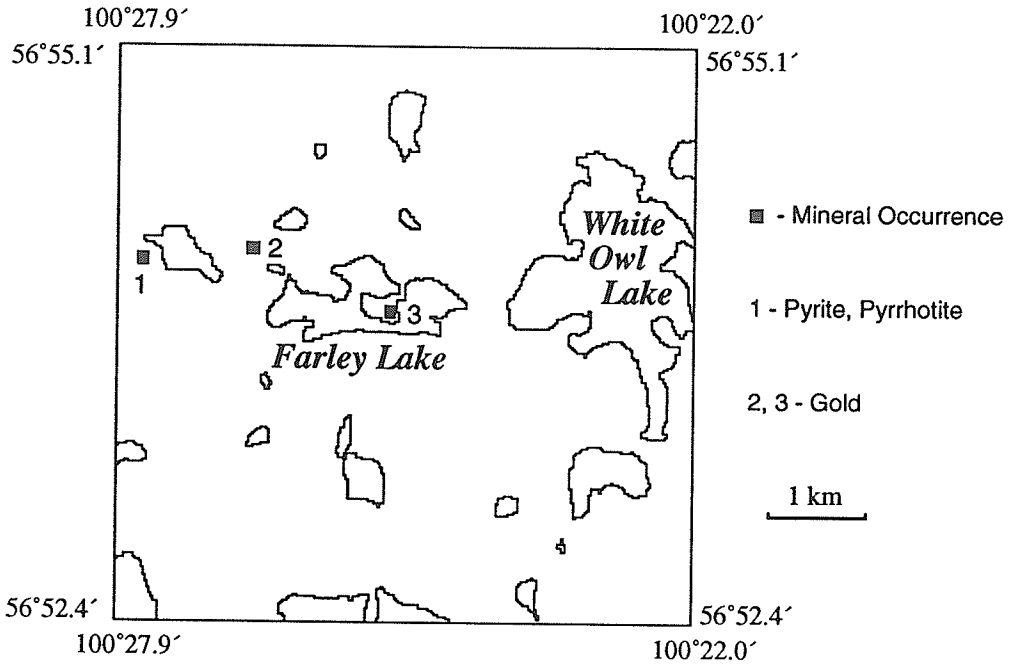

Figure 2.3: Mineral Occurrence Location Map (Farley Lake).

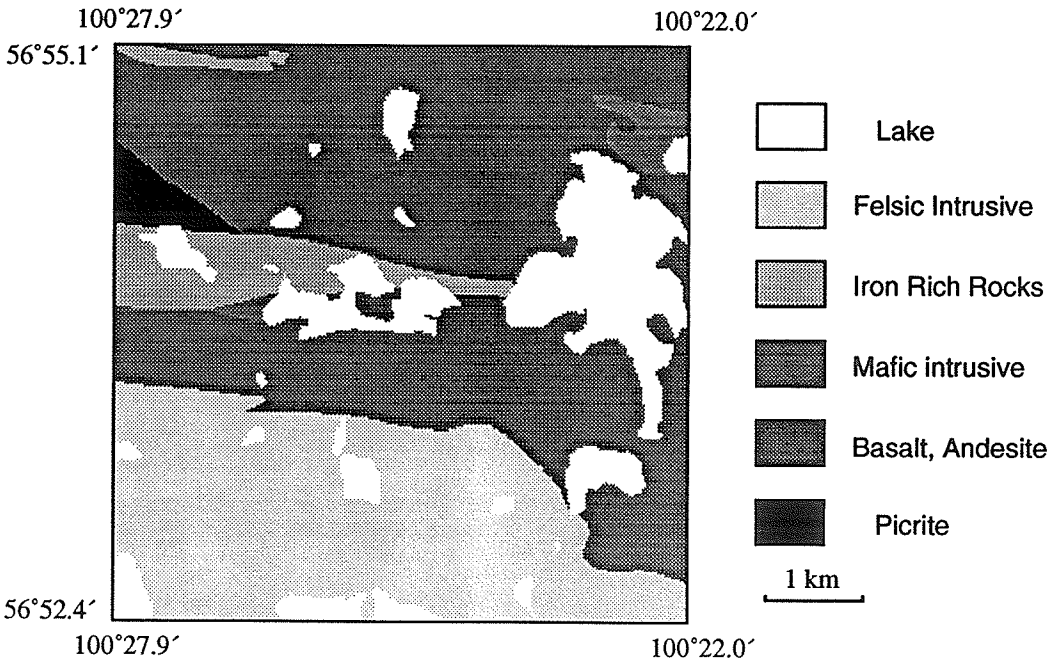

Figure 2.4: Bedrock Geology Map (Farley Lake).

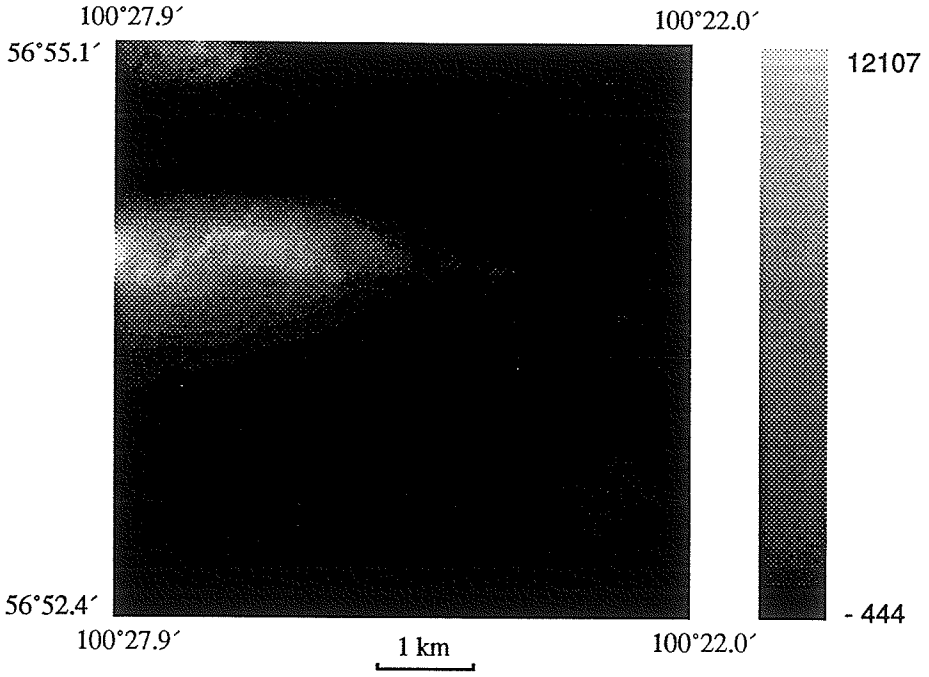

Figure 2.5: Airborne Magnetic Total Field Map (γ) (Farley Lake).

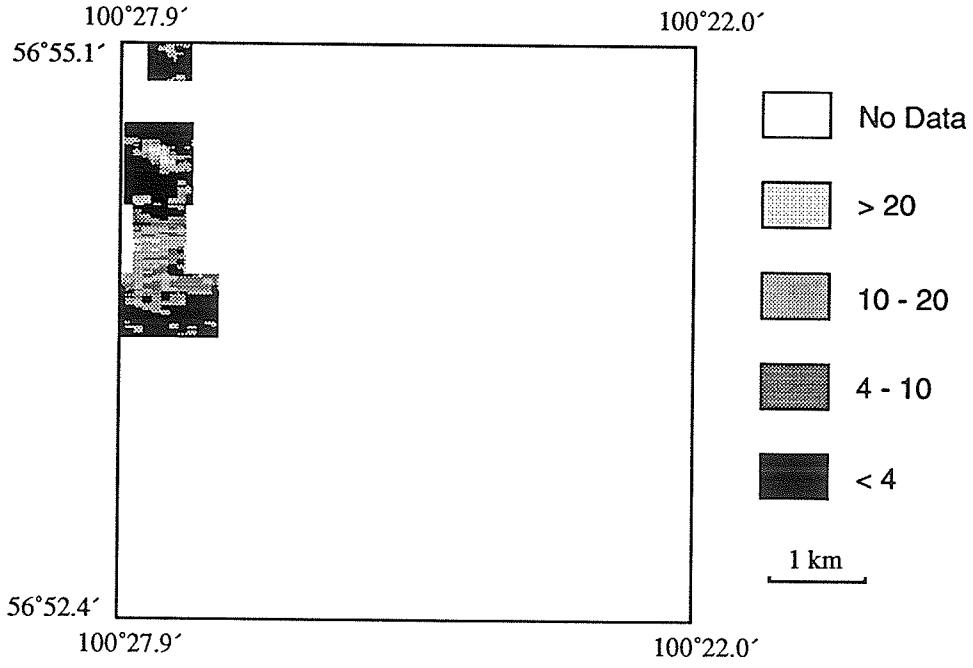

Figure 2.6: Ground Electromagnetic Map (Farley Lake).
Legend represents amplitude of the EM anomaly.

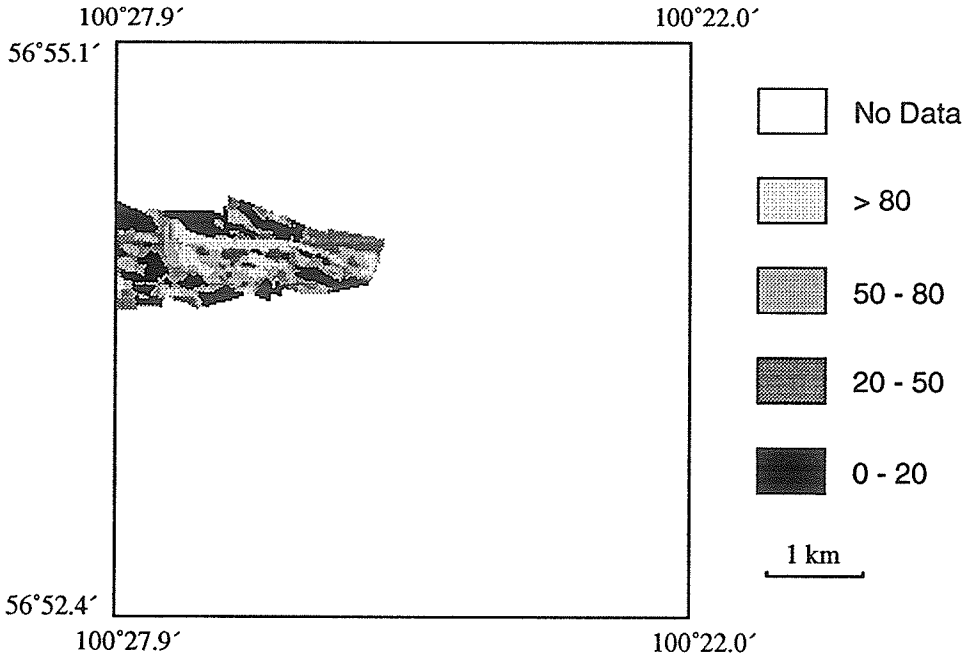

Figure 2.7: VLF Electromagnetic Contour Map (Station: NNS) (Farley Lake). Legend unit is normalized to % background.

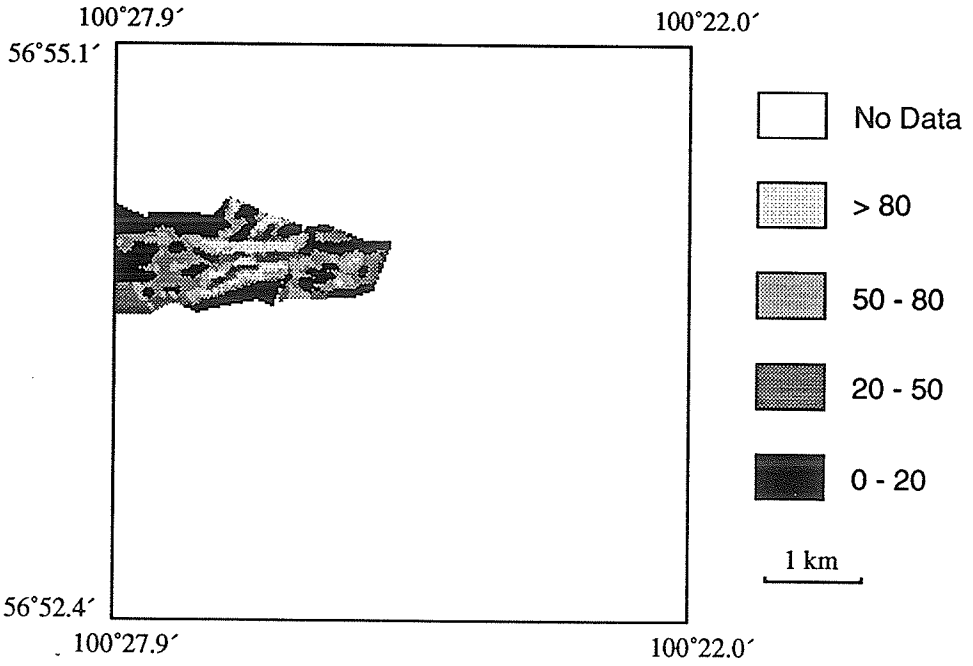

Figure 2.8: VLF Electromagnetic Contour Map (Station: NLK) (Farley Lake). Legend unit is normalized to % background.

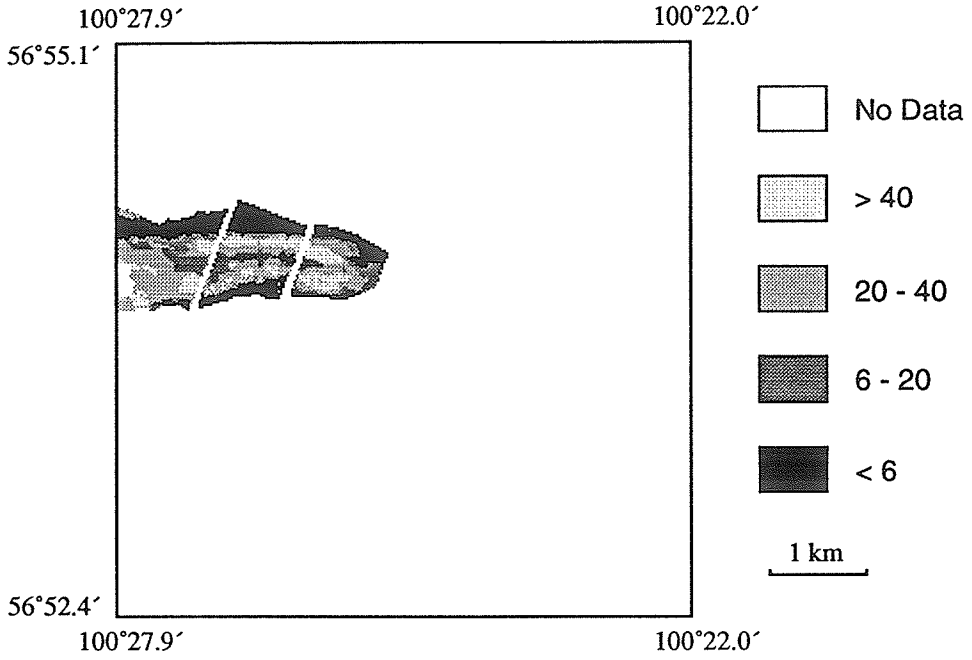

Figure 2.9: IP Chargeability Contour Map (Farley Lake).
Legend unit: mv/v.

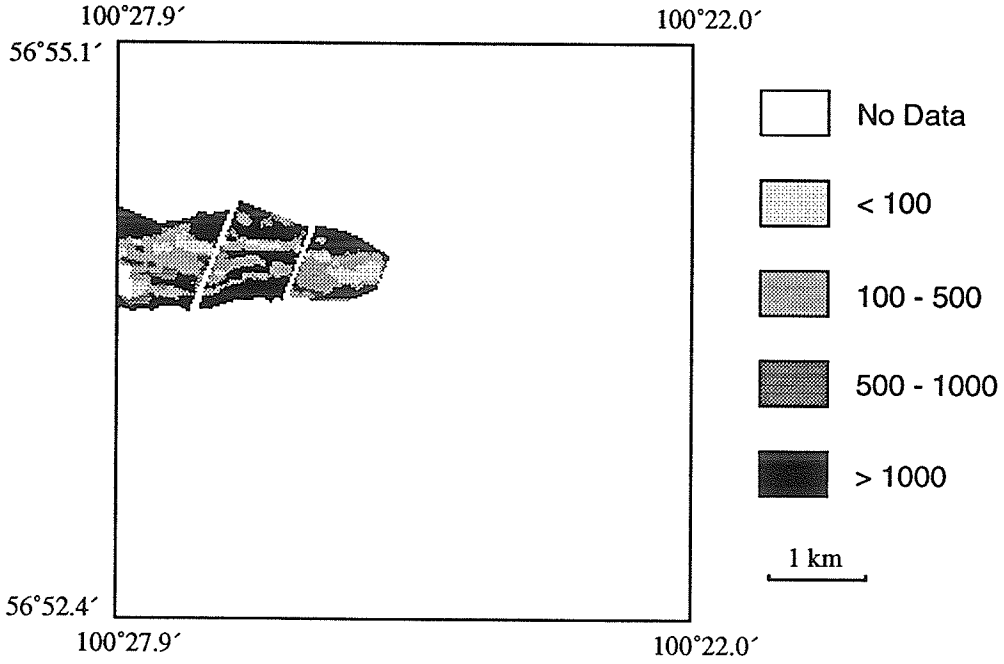

Figure 2.10: Ground Resistivity Contour Map (Farley Lake).
Legend unit: ohm-m.

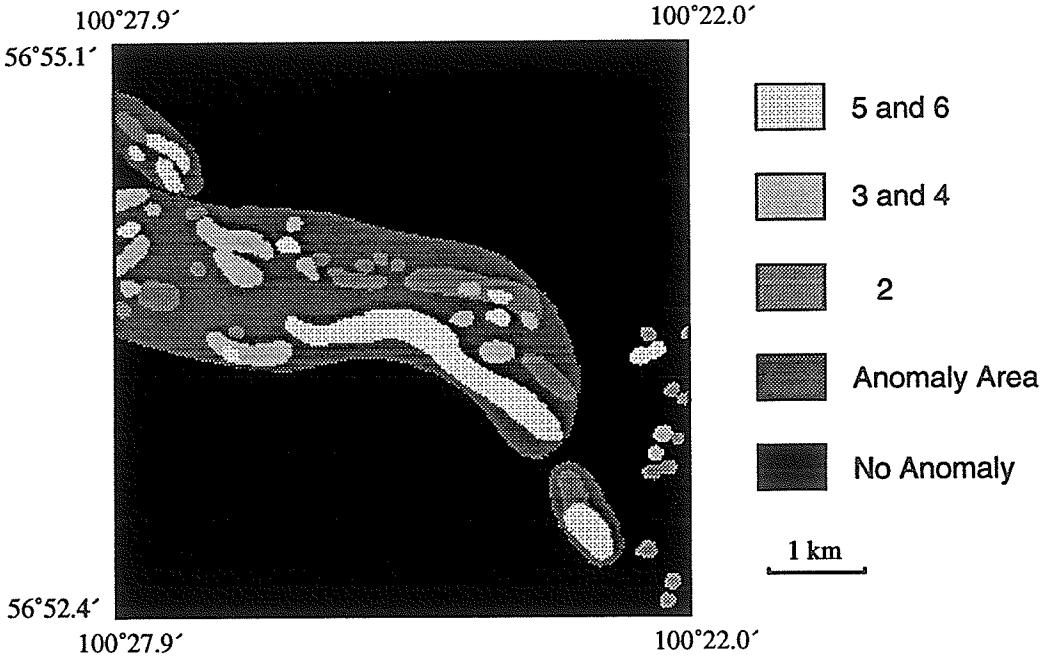

Figure 2.11: Airborne INPUT Electromagnetic Map (Farley Lake).
Legend represents number of anomaly channels. Anomaly area: low channels anomaly frequently observed.

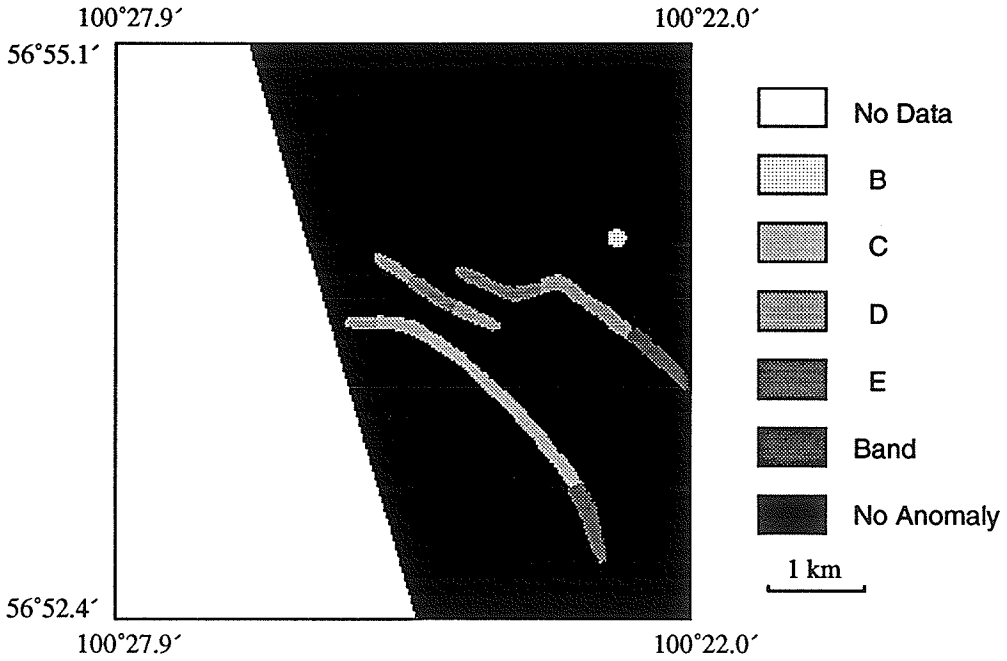

Figure 2.12: Airborne Electromagnetic Map (Farley Lake).
Legend: A, B, C, D and E represent the relative anomaly amplitude from high to low; band represents very low and wide anomaly.

relatively classified as weak, medium, and strong anomalies. The ground magnetic anomalies were compiled by Hosain (1988) from six surveyes. I re-examined original survey maps and classified them as weak, medium, and high anomalies relatively (Figure 2.17). The Airborne Electromagnetic Anomaly Map (Figure 2.16) are classified by dominant r =in-phase/out-of-phase ratio. A strong anomaly has a dominant ratio $r > 1.0$; a medium anomaly has a dominant ratio $0.8 < r \leq 1.0$; a weak anomaly $0.5 < r \leq 0.8$; a very weak anomaly $0.3 < r \leq 0.5$. $r < 0.3$ is considered as no anomaly in this case. Only lithological types are digitized from the Bedrock Geology Map (Figure 2.19). Minor re-classification had to be done before digitization because the test area covers the joint line of two maps with different lithological classifications.

Only the Geology Map (Figure 2.19) and Airborne Magnetic Total Field Map (Figure 2.15) cover the whole test area. The Ground Electromagnetic Anomaly Map (Figure 2.14), Airborne Electromagnetic Anomaly Map (Figure 2.16), and Airborne Magnetic Total Intensity Anomaly Map (Figure 2.18) cover part of the test area. The Ground Magnetic Anomaly Map (Figure 2.17) only covers a very small portion of the test area. Unfortunately, four of the six data sets do not cover the locations where the Cu-Zn deposits occur.

Map Name	Description	Source of Data
Ground Electromagnetic Anomaly Map Figure 2.14	Compiled from Open Assessment Files Scale 1:50,000	Hosain, 1988
Airborne Magnetic Total Field Map Figure 2.15	Scale 1:50,000	Hosain, 1988
Airborne Electromagnetic Anomaly Map Figure 2.16	Mean flight line spacing 660 feet Scalee 1:15,840	Fosco Mining Ltd., 1971. Manitoba Energy and Mines, Open Assessment File 92130.
Ground Magnetic Anomaly Map Figure 2.17	Compiled from Open Assessment Files Scale 1:50,000	Hosain, 1988
Airborne Total Magnetic Intensity Anomaly Map Figure 2.18	Mean flight line spacing 660 feet Scale 1:15,840	Fosco Mining Ltd., 1971. Manitoba Energy and Mines, Open Assessment File 92130
Bedrock Geology Map Figure 2.19	Preliminary Map Scale 1:50,000	Hosain, 1988

Table 2.2: Description of the Data Sets in the Snow Lake Test Area.

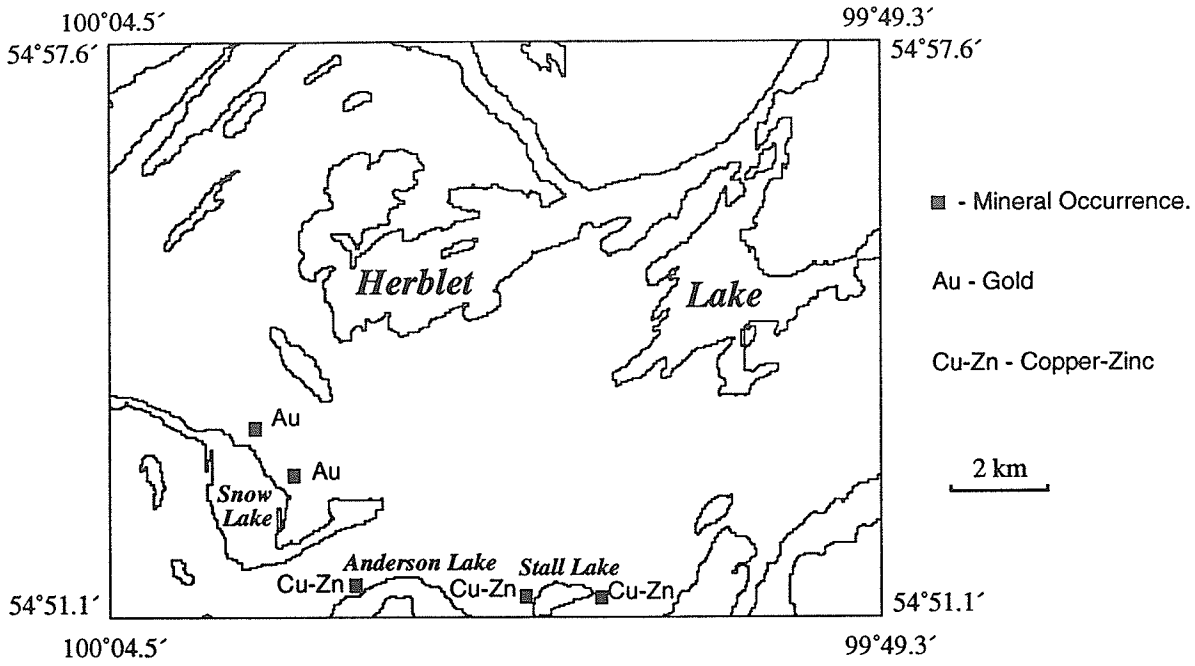

Figure 2.13: Mineral Occurrence Location Map (Snow Lake).

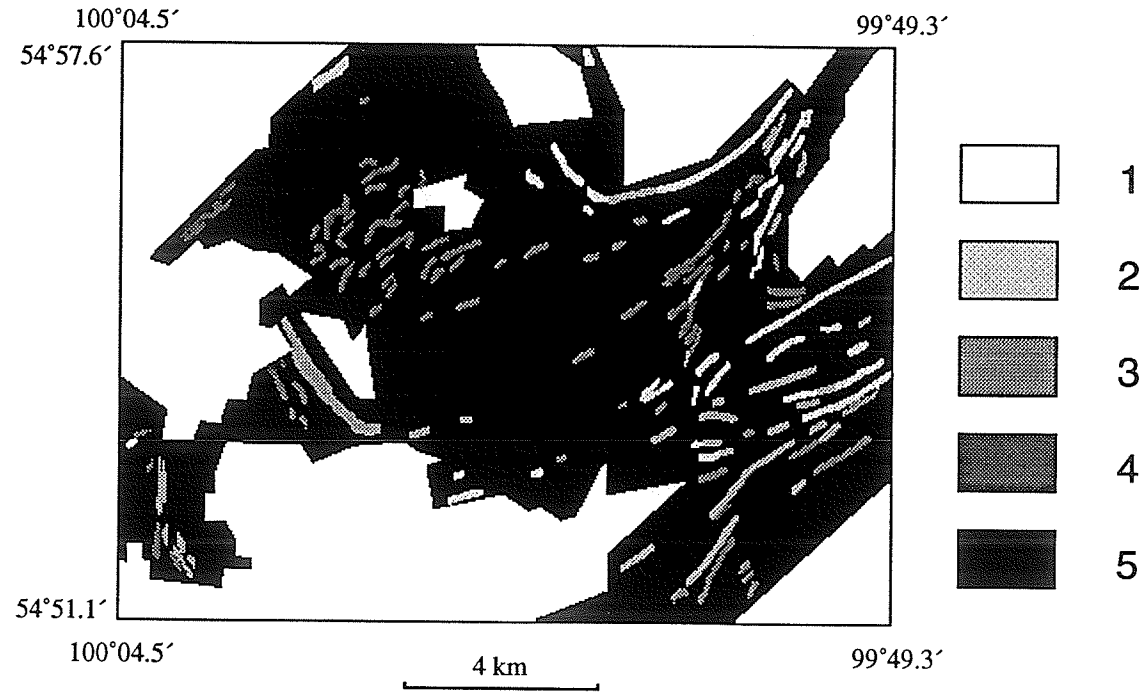

Figure 2.14: Ground Electromagnetic Anomaly Map (Snow Lake).
1. no data; 2. strong anomaly; 3. medium anomaly; 4. weak anomaly; 5. no anomaly.

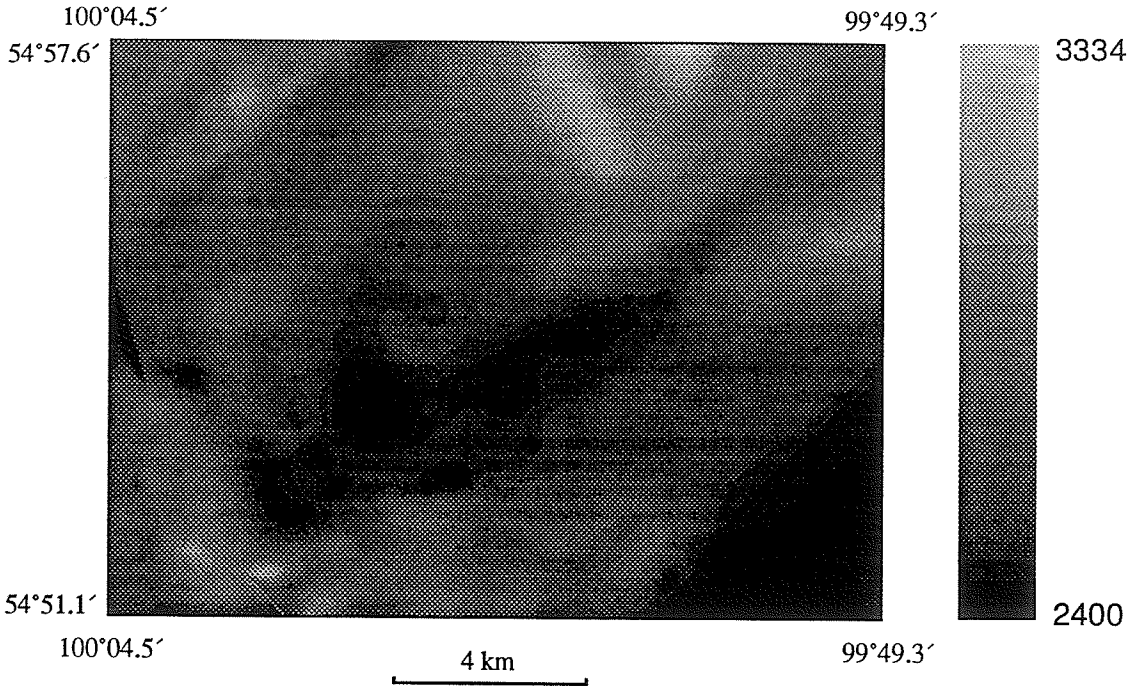

Figure 2.15: Airborne Magnetic Total Field Map (Snow Lake).

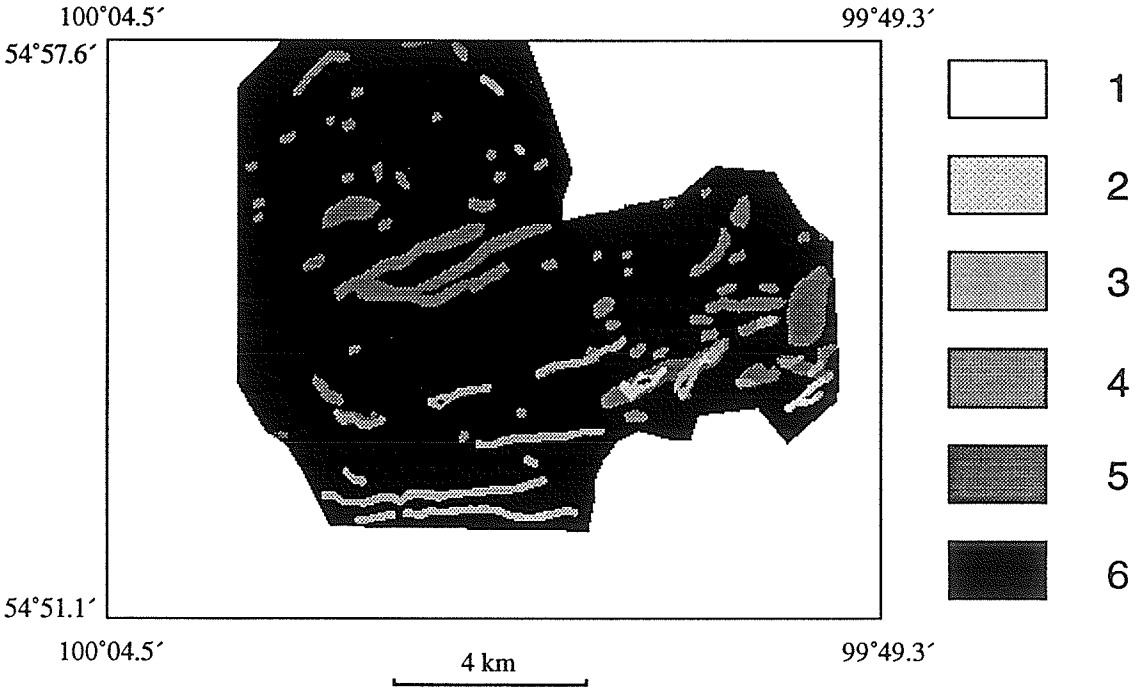

Figure 2.16: Airborne Electromagnetic Anomaly Map (Snow Lake).
1. no data; 2. strong anomaly; 3. medium anomaly; 4. weak anomaly; 5. very weak anomaly; 6. no anomaly.

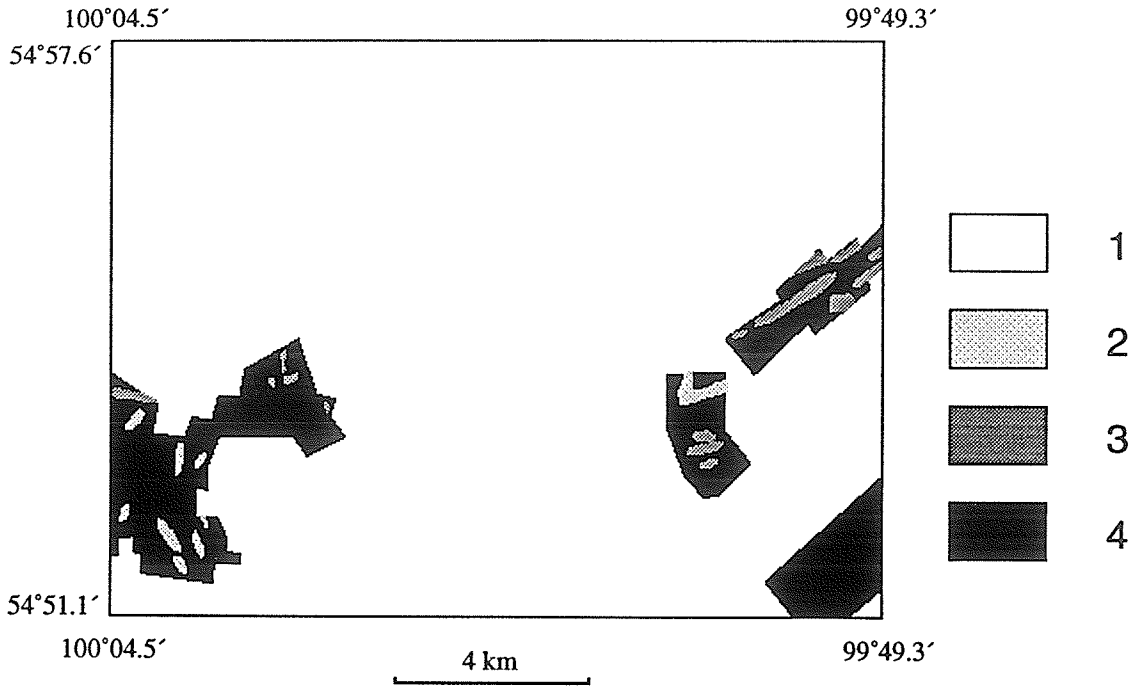

Figure 2.17: Ground Magnetic Anomaly Map (Snow Lake).
1. no data; 2. high anomaly; 3. medium anomaly; 4. low anomaly.

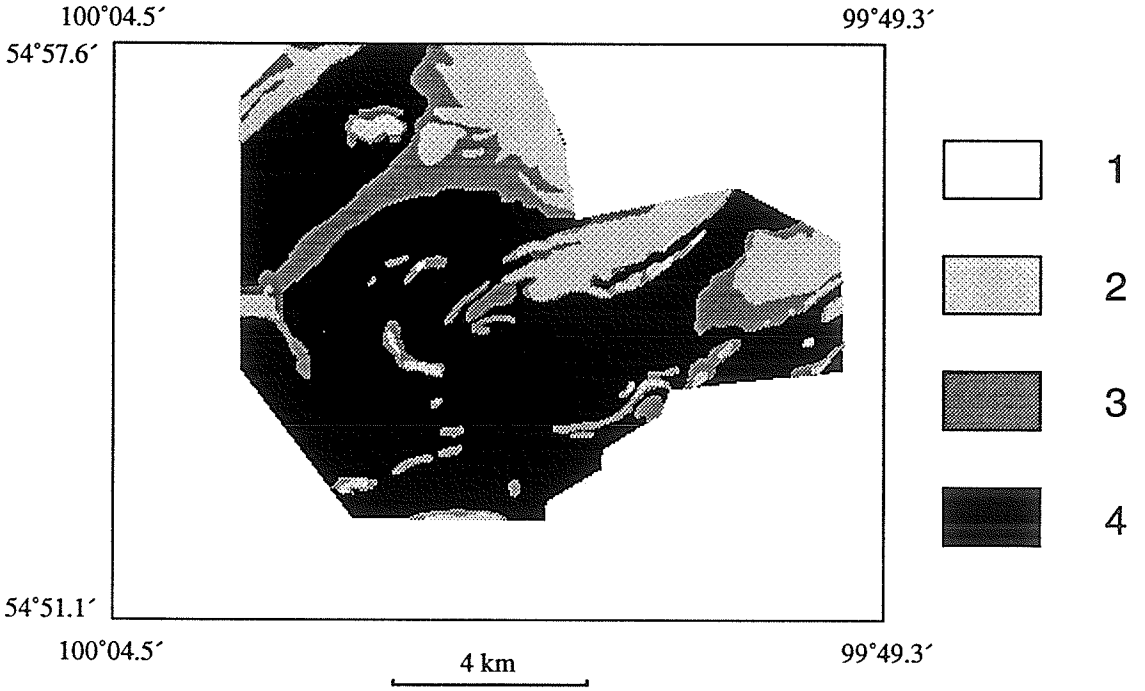

Figure 2.18: Airborne Magnetic Total Intensity Anomaly Map (Snow Lake).
1. no data; 2. high anomaly ($> 5500\gamma$); 3. medium anomaly ($5400 - 5500\gamma$); 4. low anomaly ($< 5400\gamma$).

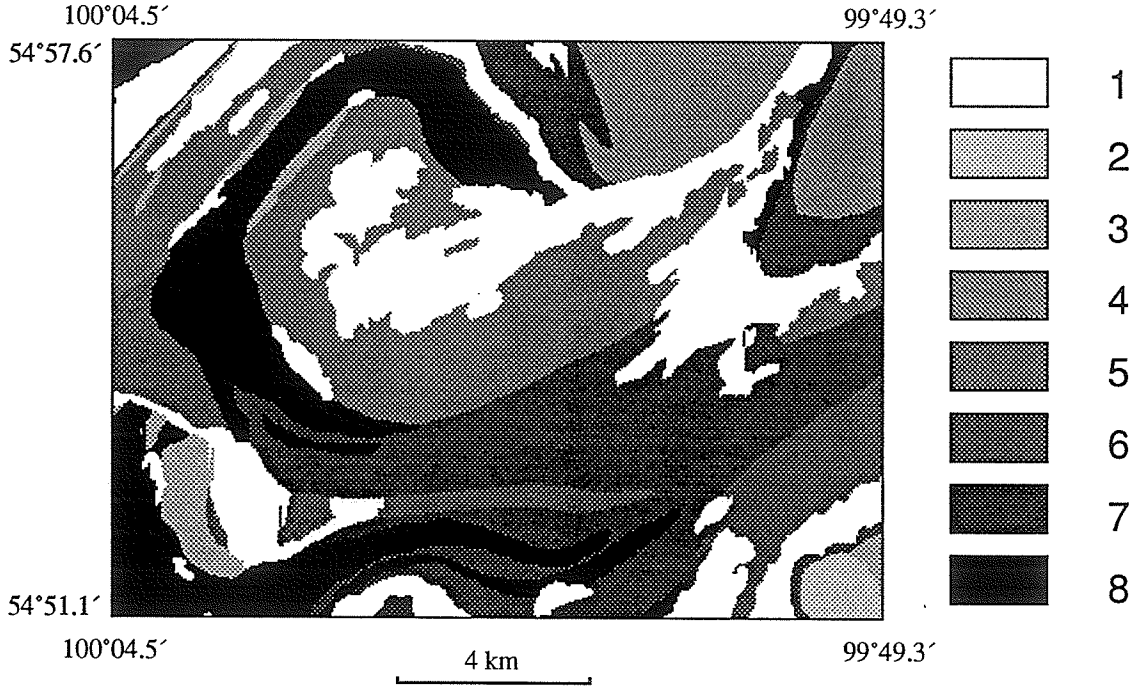

Figure 2.19: Geology Map (Snow Lake).
1. lake; 2. felsic intrusions; 3. intrusions (not differentiated); 4. quartzofeldspathic gneiss and migmatite; 5. sandstone and conglomerate; 6. greywacke, siltstone, mudstone; 7. felsic volcanic rocks; 8. mafic to intermediate volcanic rocks and related sedimentary rocks.

Chapter 3

Set-Theoretic Representation and Integration of Geoscience Data

This chapter discusses set-theoretic representation of geoscience information which in its original form exists as a collection of symbols and numbers. The first step of processing geoscience information involves representation of the survey data. Integration of different geoscience data sets can then be viewed as a special type of mapping operation. The general properties of representation and integration of geoscience data are then reviewed and studied based on the mathematical framework.

3.1 Geoscience Data Sets as Classical Sets

The classical set theory was founded by Georg Cantor (1845-1918), a German mathematician who published a series of papers over the last three decades of the 19th century. As he defined, "A set is a collection into a whole of definite, distinct objects of our intuition or our thought. The objects are called the elements (members) of the set." Clearly, this definition has loosely and imprecisely defined meaning. It makes use of the intuitive concept of the set, which is recognized as characteristics of

the "naive set theory". Mathematicians after Cantor (for example, Kuratowski and Moslowski, 1976) tried to represent the set theory as an axiomatic system by defining the set on a well-defined system of axioms. This is the so-called the axiomatic set theory.

An observation or measurement in geological sciences forms a simple set with certain specifications. A geophysical measurement, for example, usually contains instrument parameters of the observation, measurement value, attributes, location, and time of the observation. A geological observation may include rock type, age of the rock, petrographic features of the rock, location, time of the observation, and name of the observer. A point in a remote sensing image may be defined by its sensor parameters, its reflectance value, location, and time of the observation. This type of simple set can uniquely represent a scientific observation and we define it as a data object. Formally, a data object is a subset which contains all specifications that can uniquely define the observation.

Based on this definition, geoscience data sets are classical sets which contain data objects. Data set A can be written as

$$A = \{x : \varphi(x)\}$$

where x is a data object and $\varphi(x)$ are formulae or specifications which x must satisfy in order to be a member of A. It can also be written as:

$$\exists A \forall x (x \in A \leftrightarrow \varphi(x))$$

and it is called the axiom of comprehension. The inclusion relation is defined as:

$$\exists x (x \in A \rightarrow x \in B) \leftrightarrow A = B$$

Geoscience data sets defined above satisfy axioms defining a set and the laws of union, intersection, and subtraction. Let A, B and C be sets. Some of the frequently used laws in the set theory are

1. The commutative law

$$A \cup B = B \cup A;$$

$$A \cap B = B \cap A.$$

2. The associative law

$$A \cup (B \cup C) = (A \cup B) \cup C;$$

$$A \cap (B \cap C) = (A \cap B) \cap C.$$

3. The distributive law

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C);$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

4. De Morgan's law

$$A - (B \cap C) = (A - B) \cup (A - C);$$

$$A - (B \cup C) = (A - B) \cap (A - C).$$

As an example, let the following set be defined as:

$$M_a = \{x : \text{magnetic data } \} ;$$

$$M_b = \{x : \text{magnetic data in the area } \Psi\} ;$$

$$M = \{x : \text{geophysical data } \} .$$

Then we have

$$M_b \subseteq M_a \subseteq M;$$

$$M_a \cap M_b \cap M = M_b;$$

$$M_a \cup M_b \cup M = M.$$

A set of data objects, as a classical set, generally has clearly defined boundaries although the data itself might be imprecise or even wrong. A data object can either be a member of a set with a membership 1 or not a member with a membership 0.

For resource evaluation or exploration purposes, a data structure should be defined. Let us define a D-set as a set of data objects which represent the same physical or chemical property and only have one spatial coverage. This definition means that data from different surveys can be calibrated and compiled to form a D-set if they are measurements of the same physical or chemical property and they do not overlap spatially. The data should be organized as different D-sets if they spatially overlap and they are measurements of the same property. For instance, two magnetic surveys made at different times over an area are treated as two D-sets; remote sensing images from different channels form different D-sets. A lithological map is a D-set and a geological structure map forms another. A D-set can be derived from another D-set or other D-sets.

Let us define an exploration set X, which is called X-set, as a collection of all D-sets which have direct or indirect implications to an exploration target in a defined area. Given an X-set

$$X = \{D_1, D_2, \ldots, D_n\}$$

where D_i $(i = 1, 2, \ldots, n)$ are D-sets. Any X-set is an essential part of an exploration project. The exploration activities discussed below are based on the X-set.

3.2 G-mapping Operations

In geological research and exploration problems, we have to absorb information from several D-sets to reach a subset of results and we usually have to consider subsets

in a larger area for a decision at a specific location. In some cases, specific mapping operations are required to represent the D-sets into a fuzzy space or a probability space in which they can be processed more accurately and effectively. A mapping operation, G-mapping, is defined for describing the data operations in geoscience problems, particularly in integrated resource exploration problems.

Definition: G-mapping:

Let

$$A = \{A_1, A_2, \ldots, A_n\}$$

$$B = \{B_1, B_2, \ldots, B_m\}$$

$$C = \{C_1, C_2, \ldots, C_l\}$$

be sets of classical objects, fuzzy objects, or probability objects, where A_i $(i = 1, 2, \ldots, n)$ is a subset of A; B_i $(i = 1, 2, \ldots, m)$ is a subset of B; C_i $(i = 1, 2, \ldots, l)$ is subset of C. Suppose G is a set of relations; then it is called a G-mapping if and only if $B = G(A)$ and $C = G(A)$ imply that $B = C$. We say that G is a G-mapping of A into B and it can be denoted as

$$G : A \rightarrow B$$

or

$$A \xrightarrow{G} B.$$

This definition allows mapping operations among the classical sets, fuzzy sets, and probability sets and it encompasses fuzzy operations in fuzzy space and probability operations in probability space. Some special cases of G-mapping are discussed below.

Let

$$A = \{A_1, A_2, \ldots, A_n\}$$

$$B = \{B_1, B_2, \ldots, B_m\}$$

be sets, where A_i $(i = 1, 2, \ldots, n)$ is a subset of A and B_i $(i = 1, 2, \ldots, m)$ is a subset of B. And

$$A_i = \{a_i^1, a_i^2, \ldots, a_i^k\}$$

$$B_i = \{b_i^1, b_i^2, \ldots, b_i^h\}$$

where a_i^j $(j = 1, 2, \ldots, k; i = 1, 2, \ldots, n)$ is an object in A_i and b_i^j $(j = 1, 2, \ldots, h; i = 1, 2, \ldots, m)$ is an object in B_i. G is a G-mapping of A into B or $G : A \rightarrow B$.

Case 1. $n = m = k = h = 1$

$$B = G(A) = G(\{a\}) = \{b\}$$

This is a single data point transform.

Case 2. $n = m = 1$, $k = h$, and for every $a \in A$ we have exactly one b such that

$$B_1 = \{G(\{a\}) \mid a \in A_1\}$$

This is a traditional binary mapping.

Case 3. $n = m = 1$.

$$B_1 = \{G(A_1^i) \mid A_1^i \subseteq A_1\}.$$

where $i = 1, 2, \ldots, h$. In this case, a subset A_1^i of A_1 is used as input to determine an object in B_1.

Case 4. $n > 1$, $m = 1$.

$$B_1 = \{G(A_1^i, A_2^i, \ldots, A_n^i) \mid A_1^i \subseteq A_1, A_2^i \subseteq A_2, \ldots, A_n^i \subseteq A_n\}$$

where $i = 1, 2, \ldots, h$. This G-mapping combines several subsets to form a new subset.

Case 5. $m > 1$, $n > 1$ and $G = \{G_1, G_2, \ldots, G_m\}$

$$B = \left\{ \begin{array}{l} B_1 \\ B_2 \\ \vdots \\ B_m \end{array} \right\} = \left\{ \begin{array}{l} \{G_1(A_1^i, A_2^i, \ldots, A_n^i)\} \\ \{G_2(A_1^i, A_2^i, \ldots, A_n^i)\} \\ \vdots \\ \{G_m(A_1^i, A_2^i, \ldots, A_n^i)\} \end{array} \right\}$$

where $A_1^i \subseteq A_1$, $A_2^i \subseteq A_2$, ..., $A_n^i \subseteq A_n$; $i = 1, 2, \ldots, h$. This case corresponds to operations on subsets to create new subsets for different needs.

Case 6. $m > 1$, $n > 1$, and $G = \{G_1, G_2, \ldots, G_n\}$

$$B = \begin{Bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{Bmatrix} = \begin{Bmatrix} \{G_1(A_1^i)\} \\ \{G_2(A_2^i)\} \\ \vdots \\ \{G_n(A_n^i)\} \end{Bmatrix}$$

where $A_1^i \subseteq A_1$, $A_2^i \subseteq A_2$, ..., $A_n^i \subseteq A_n$; $i = 1, 2, \ldots, h$. This represents the operations which map separately all subsets inside A into B.

Two specific types of G-mappings, $X - F$ which maps classical sets into fuzzy space and $X - P$ which maps classical sets into probability space, can be defined as below.

Definition: $X - F$ mapping. Let A be a set of classical objects. A G-mapping G is an $X - F$ mapping if and only if $G : A \rightarrow F_1$ and $G : A \rightarrow F_2$ imply that $F_1 = F_2$, where F_1 and F_2 are fuzzy subsets or sets of fuzzy objects. We say that the $X - F$ mapping transforms a classical set into a fuzzy set or fuzzy space.

Definition: $X - P$ mapping.

Let A be a set of classical objects. A G-mapping G is an $X - P$ mapping if and only if $G : A \rightarrow P_1$ and $G : A \rightarrow P_2$ imply that $P_1 = P_2$, where P_1 and P_2 are probability sets or sets of probabilities. We say that the $X - P$ mapping transforms a set into a probability set or probability space.

The G-mapping defined above applies to the operations in both fuzzy space and probability space because it is a generalized description of mathematical operations with details ignored. The only requirement for a mathematical operation to be a G-mapping is that the value from the operation will not change if the variables for the operation are not changed. It must be satisfied by all mathematical operations.

The equivalence of G-mappings G_1 and G_2 is defined as

$$\bigwedge_{A_i \subseteq A} (G_1(A_i) = G_2(A_i)) \leftrightarrow G_1 = G_2.$$

Let A and B be sets and let G_1 and G_2 be G-mappings

$$\{(A, B) : \bigvee_{C} (A \xrightarrow{G_1} C \wedge C \xrightarrow{G_2} B)\}$$

which is called the composition of G_1 and G_2 and denoted as $G_1 \circ G_2$.

Let G_1, G_2, G_3, and G_4 be mappings. If

$$\bigwedge_{A_i \subseteq A} (G_1 \circ G_2(A_i) = G_3 \circ G_4(A_i)) \leftrightarrow G_1 \circ G_2 = G_3 \circ G_4$$

we say that $G_1 \circ G_2$ and $G_3 \circ G_4$ commute.

Suppose G is a G-mapping. G is pairwise if and only if there exists a function g, such that

$$G(A_1, A_2, \ldots, A_n) = g(\ldots g(g(A_1, A_2), A_3) \ldots, A_n)$$

where g is called its pairwise function. The pairwise property allows G to be calculated step by step.

Let G be a G-mapping and g its pairwise function. If

$$g(g(A_i, A_j), A_k) = g(A_i, g(A_j, A_k)),$$

where $i, j, k, \leq n$, we say that G is symmetric. This property allows $A_1, A_2, \ldots, A_n$ to be combined one after another without concerning the order in which they are combined.

3.3 Integration as Special Types of G-mappings

The first task one has to do prior to an integrated resource exploration or evaluation project is to establish a database relevant to the target in the prospect region. An

exploration set or X-set which contains several D-sets of data objects can be obtained by structuring or organizing the database. Suppose

$$X = \{D_1, D_2, \ldots, D_n\}$$

is an X-set and

$$T = \{T_1, T_2, \ldots, T_m\}$$

is a set of exploration targets or propositions associated with the exploration targets, and

$$G = \{G_1, G_2, \ldots, G_m\}$$

is a set of G-mappings which maps X into T. Then we have

$$T = \left\{ \begin{array}{l} T_1 \\ T_2 \\ \vdots \\ T_m \end{array} \right\} = \left\{ \begin{array}{l} \{G_1(D_1^i, D_2^i, \ldots, D_n^i)\} \\ \{G_2(D_1^i, D_2^i, \ldots, D_n^i)\} \\ \vdots \\ \{G_m(D_1^i, D_2^i, \ldots, D_n^i)\} \end{array} \right\}$$

where $D_1^i \subseteq D_1$, $D_2^i \subseteq D_2$, $\ldots$, $D_n^i \subseteq D_n$; $i = 1, 2, \ldots, h$. This means that an integrated resource exploration is nothing more than a G-mapping of X into T, i.e. $G : X \rightarrow T$. But unfortunately, we do not know or have not had generalized analytical relations of these G-mappings. Most efforts related to the resource exploration fall into two categories based on the above description. In the first are those which attempt to improve an X-set, ranging from data acquisition to processing. The backbone of most data management and processing in this category, either numeric (mostly in geophysics and remote sensing) or symbolic (mostly in geology), is the classical set theory although the actual data objects have been intrinsically imprecise. The efforts trying to establish and improve the G-mapping of X into T fall into the second category. In this category, explorationists have to face the impreciseness of the data objects, incompleteness of spatial data coverage, and lack of enough knowledge. They

have to process these deficiencies as accurately as possible. It is almost impossible to establish T on the basis of the classical set theory because an explorationist can never simply answer "yes" or "no" about a prospect until a well is drilled and the final assessment is made. This thesis research is more focused on the second category.

The following is an example which shows that there are different procedures in data integration approaches and that uncertainties in this type of problems require fuzzy logic and probabilistic representation and operations.

Suppose we have D-sets in an area Ψ:

$$D_1 = \{x : \text{magnetic data }\},$$

$$D_2 = \{x : \text{airborne EM data }\},$$

$$D_3 = \{x : \text{geological data }\}$$

then we have an X-set

$$X = \{D_1, D_2, D_3\}.$$

Let T_B be a target subset or target proposition for base metal deposits. The task here is to map X into T_B, ie. $X \xrightarrow{G} T_B$. This G-mapping can be decomposed into steps. By interpreting D_1, a new evidence set can be created:

$$E_1 = \{e : \text{base metal deposits exist from } D_1\}$$

and similarly, the other two D-sets can be represented as evidence set

$$E_2 = \{e : \text{base metal deposits exist from } D_2\}$$

$$E_3 = \{e : \text{base metal deposits exist from } D_3\}$$

and then we have

$$E = \{E_1, E_2, E_3\}.$$

These mappings represent a type of G-mapping, Case 6, described in **3.2**. And then,

E_1, E_2, and E_3 can be integrated by a G-mapping

$$G : E \longrightarrow T_B$$

The whole procedure can be sketched as

$$\left.\begin{array}{l} D_1 \longrightarrow E_1 \\ D_2 \longrightarrow E_2 \\ D_3 \longrightarrow E_3 \end{array}\right\} \xrightarrow{G} T_B.$$

An alternative procedure can be taken for the same task. Such as

$$\left.\begin{array}{l} \left.\begin{array}{l} D_1 \rightarrow E_1 \\ D_2 \rightarrow E_2 \end{array}\right\} \rightarrow E_4 \\ D_3 \longrightarrow E_3 \end{array}\right\} \longrightarrow T_B$$

where E_4 in the sketch is a set by combining E_1 and E_2 and represents evidence for base metal deposits from geophysical data D_1 and D_2 in the area. E_4 is then combined with E_3 which represent the evidence from geological data D_3.

The above example shows that there are different approaches and they can be represented by G-mappings. Apparently, there are great difficulties to establish E_i $(i = 1, 2, 3, 4)$ and T_B in a framework of the classical set theory. A classical set has clearly defined boundaries. An object can either be a member with a membership 1 or not a member with a membership 0. No other situation is allowed. A more natural approach is to map the exploration information into a fuzzy space or probability space with $X - F$ or $X - P$ mappings. And then they can be processed using fuzzy operations in the fuzzy space or probability rules in the probability space. The above example can be sketched as

$$\left.\begin{array}{l} D_1 \xrightarrow{X-F} F_1 \\ D_2 \xrightarrow{X-F} F_2 \\ D_3 \xrightarrow{X-F} F_3 \end{array}\right\} \xrightarrow{FO} T_B$$

in fuzzy space and

$$\left.\begin{array}{l} D_1 \xrightarrow{X-P} P_1 \\ D_2 \xrightarrow{X-P} P_2 \\ D_3 \xrightarrow{X-P} P_3 \end{array}\right\} \xrightarrow{PO} T_B$$

or

$$\left.\begin{array}{l} \left.\begin{array}{l} D_1 \xrightarrow{X-P} P_1 \\ D_2 \xrightarrow{X-P} P_2 \end{array}\right\} \xrightarrow{PO} P_4 \\ D_3 \xrightarrow{X-P} P_3 \end{array}\right\} \xrightarrow{PO} T_B$$

in probability space, where FO is a fuzzy operator, and PO a probability operator, respectively.

3.4 Integration Using Bayes' Rule

As an example of set-based representation and integration of exploration data, a Bayesian probability approach is represented as G-mappings and some of the important features are discussed below. Suppose

$$X = \{D_1, D_2, \ldots, D_n\}$$

is an X-set, where $D_1, D_2, \ldots, D_n$ are D-sets.

Let P be a set which contains the probabilities of all possible occurrences (events) implied by X.

$$P = \{G(D_1^i, D_2^i, \ldots, D_n^i)\},$$

where G is an $X - P$ mapping and $D_1^i \subseteq D_1, D_2^i \subseteq D_2, \ldots, D_n^i \subseteq D_n$. $i = 1, 2, \ldots, h$. h is the maximum number of possible events. It can be sketched as

$$\left.\begin{array}{l} D_1 \\ D_2 \\ \vdots \\ D_n \end{array}\right\} \xrightarrow{G} P.$$

Let T_t be a set of specific target t. According to Bayes' rule, we have

$$P(T_t|a_1^i \cap a_2^i \ldots \cap a_n^i) = \frac{P(T_t)P(a_1^i \cap a_2^i \ldots \cap a_n^i|T_t)}{P(a_1^i \cap a_2^i \ldots \cap a_n^i)} \tag{3.1}$$

and under the assumption of conditional independence,

$$P(T_t|a_1^i \cap a_2^i \ldots \cap a_n^i) = \frac{P(T_t)P(a_1^i|T_t)P(a_2^i|T_t)\ldots P(a_n^i|T_t)}{P(a_1^i \cap a_2^i \ldots \cap a_n^i)} \tag{3.2}$$

where $P(T_t)$ in this formula can be treated as a constant in a chosen exploration area. A favorability can then be defined as

$$P_f = \frac{P(T_t|a_1^i \cap a_2^i \ldots \cap a_n^i)}{P(T_t)} \tag{3.3}$$

or

$$P_f = \frac{P(a_1^i \cap a_2^i \ldots \cap a_n^i|T_t)}{P(a_1^i \cap a_2^i \ldots \cap a_n^i)}. \tag{3.4}$$

The P_f in Eq. (3.4) is called the favorability function. Under the assumption of conditional independence,

$$P_f = \frac{P(a_1^i|T_t)P(a_2^i|T_t)\ldots P(a_n^i|T_t)}{P(a_1^i \cap a_2^i \ldots \cap a_n^i)}. \tag{3.5}$$

Theorem: If $A_1, A_2, \ldots, A_n$ are independent, the favorability function is symmetrical.

Proof: According to the assumption of independence, we have

$$P(a_1^i \cap a_2^i \ldots \cap a_n^i) = P(a_1^i)P(a_2^i)\ldots P(a_n^i) \tag{3.6}$$

and then

$$\begin{aligned} P_f &= \frac{P(a_1^i|T_t)}{P(a_1^i)}\frac{P(a_2^i|T_t)}{P(a_2^i)}\cdots\frac{P(a_n^i|T_t)}{P(a_n^i)} \\ &= p_1^i p_2^i \ldots p_n^i \end{aligned}$$

where $p_j^i = \frac{P(a_j^i|T_t)}{P(a_j^i)}$ $(j = 1, 2, \ldots, n)$. The pairwise function

$$g(p_j^i, p_{j+1}^i) = p_j^i p_{j+1}^i$$

and clearly

$$g(g(p_j^i, p_{j+1}^i), p_{j+2}^i) = g(p_j^i, g(p_{j+1}^i, p_{j+2}^i)).$$

Hence the theorem holds.

If we call p_j^i as partial favorability from D_j, the total favorability of the X-set is the product of all partial favorabilities when all D-sets are independent. Equations (3.1) and (3.4) are generally not pairwise even though they are under the conditional independence assumption.

The whole procedure of integration using the Bayes' rule can be sketched as

$$\left.\begin{matrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{matrix}\right\} \xrightarrow{X-P} P \xrightarrow{BR} \left\{\begin{matrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{matrix}\right\}$$

where BR is Bayes' rule and T_i $(i = 1, 2, \ldots, m)$ is a set of ith targets under consideration.

3.5 Comment and Discussion

Geoscience data sets can be represented as classical sets. A data structure, an X-set, can be established by restructuring or reorganizing a data base. Integrated resource exploration can be represented as a set of G-mappings based on the X-set. This representation allows the integrated resource exploration problem to be examined in a mathematically precise way. This representation, in addition, provides a mathematical basis for further advanced study of integrated exploration problems.

Chapter 4

Fuzzy Set Theory and Integration of Geological and Geophysical Data Sets

There have recently been several reported applications of fuzzy set theory in remote sensing and other geographical information processing. Among them are Wang (1989) who applied fuzzy set theory in an expert system for remote sensing image analysis and Blonda et al. (1989) who used a fuzzy logic technique in classifying multi-temporal remotely sensed imagery. This theory can also be used as a basis for integrating geoscience information (An et al., 1991).

Let us consider a subset of gravity data objects, which is defined as $A =$"Bouguer gravity anomaly of 31.52 mgal". One can define this as a subset with only 31.52 mgal. However, it is known that it is not practical to create an infinite number of subsets to represent certain information. A more logical approach would be to define a subset such that survey data from a sensor would be included together. The elements in this subset can then have varying degrees of information content as well as varying degrees of uncertainty. Secondly, the subset of a geological map can be defined as $B =$"felsic intrusives". This spatial subset of a geological map is consists of symbolic elements

but the data elements have uncertainties such as boundary errors, uncertainties in identifying the rock, scale and drawing errors on a map. Similarly, most geological, geophysical and remote sensing information cannot be precisely represented using classical set theory, either in temporal or spatial domains.

Suppose that an explorationist is searching for favorable locations for base metal deposits in an exploration area, and he/she has an aeromagnetic map. Analysis of the map will provide a subset of several anomalies, some of which may be indicative of base metal deposits. However, physical size, shape and degree of magnetic induction associated with each anomaly may be more appropriately represented by some other method than the classical set approach. Uncertainties involved in interpretation of each anomaly and of correlating the results with actual physical parameters representative of a base metal pose problems. If there are more than one data set, anomalies and/or evidence indicative of a base metal deposit with varying degree of uncertainties may be identified from different data sets at the same or at different locations. One of the first tasks of today's integrated exploration includes representation of the information from each data set using a mathematically acceptable tool and an appropriate technique of information combination for the evidence represented from data sets so that a reasonable and accurate evaluation can be obtained. Fuzzy set theory provides a suitable precise method for such cases of representing the information content of different data sets and combining them with chosen processing operations.

As mentioned above, data integration involves operations of combining information from many different D-sets. The result of information integration provides an estimate of favorability towards a specific exploration target. The concern here is however whether the final estimate is unreasonably biased towards a specific D-set. It might also be biased by an improper combination operation or by erroneous data. Quenouille (1956) introduced a technique which can be used to estimate the operational bias by splitting the given data into n groups if the data are composed of n

independent pieces. The jackknife variance can then be calculated to help evaluate the estimated result (Miller, 1968).

In this chapter, fuzzy set theory is briefly reviewed and integration of different geoscience data sets using fuzzy logic are investigated. The approach is tested using exploration examples from Farley Lake and Snow Lake Areas, Manitoba, Canada.

4.1 Fuzzy Set Theory

The fuzzy set theory was systematically formulated as early as 1965 by Zadeh (1965). A fuzzy set F is a set of ordered pairs:

$$F = \{(x, \mu_F(x)) | x \in A\}. \tag{4.1}$$

where A is a collection of objects and $\mu_F(x)$ is called the membership function or degree of compatibility of x in F; $\mu_F(x)$ maps A to the membership space. The range of $\mu_F(x)$ is usually, but not necessarily, defined in $[0, 1]$, where 0 expresses non-membership and 1 full membership. The membership function is numerically equal to the possibility distribution, π_F, for the proposition "x is F_R" if F_R acts as a fuzzy restriction associated with x (Zadeh, 1978). The concept of possibility is, in general, very different from the concept of traditional probability. For a simple example, consider a fuzzy set

$$F = \{x \text{ eggs Ping eats for breakfast}\}$$

where x takes value from $\{0, 1, 2, \ldots\}$. A possibility distribution, $\pi(x)$, can be associated with x by interpreting $\pi(x)$ as the degree of ease with which Ping can eat x eggs. $\pi(x)$ can be assessed by considering his age, weight, type of work he is doing, his former roommate's statement that Ping used to buy a dozen eggs per week, and other explicit and implicit criteria. A probability distribution, $P(x)$, may also be associated with x by interpreting $P(x)$ as the probability of Ping eating x eggs for breakfast. It

may be obtained by collecting samples for a period and statistical analysis on these samples. They might be as shown in Table 4.1

x	0	1	2	3	4	5	6	7	8
$\pi(x)$	1.0	1.0	1.0	1.0	0.80	0.50	0.30	0.10	0.0
$P(x)$	0.05	0.10	0.70	0.10	0.05	0.0	0.0	0.0	0.0

Table 4.1: The possibility and probability associated with x

The first thing to notice in this example is that $\sum \pi(x) \neq 1$ while $\sum P(x) = 1$. It should also be noticed that a high degree of possibility does not imply a high probability, nor a low probability imply a low possibility. However, if an event is impossible, it is bound to be improbable. This heuristic connection is stated as possibility/probability consistency principle and the degree of consistency of the possibility distribution $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ with probability distribution $P = \{p_1, p_2, \ldots, p_n\}$ is expressed by

$$\delta = \pi_1 p_1 + \pi_2 p_2 + \ldots + \pi_n p_n$$

(Zadeh, 1978). This is an approximate formalization of the heuristic observation that decreasing possibility of an event tends to decrease its probability of its occurrence. In another mathematical example, a fuzzy membership function representing the case of "a number x is close to 10" can be written as

$$F = \{(x, \mu_F(x)) : \mu_F = (1 + (x - 10)^2)^{-1}\}.$$

This formula represents the possibility that the number x is a real number close to 10 and it does not represent a probability distribution even though the function itself has the appearance of a probability distribution function.

4.2 Fuzzy Representation

In the framework of fuzzy set theory, the subsets discussed above may be redefined depending on the problem to be solved. For the subset F_b ="Bouger anomalies of 31.52 $mgal$", a membership can be attached to each observation according to its information content. For the subset F_f ="felsic intrusive", a membership can be given according to the certainty of identifying the rock and/or according to the strength of the evidence which supports that mapped outcrop is a felsic intrusive. Similarly, an explorationist can assign a membership to each magnetic anomaly according to the accuracy of the data and the interpreter's expertise.

In the case of resource exploration, the final objective is to generate sets of different resource types, such as base metal and iron formation in mineral exploration, and oil or gas reservoirs in petroleum exploration. For example, a survey data set for the base metal deposits contains evaluation of base metal occurrences spatially distributed in the exploration area. It is, however, almost impossible to establish this set if one completely follow the classical set theory because an explorationist can never simply say "yes" or "no" before they are drilled or observed directly. A set of spatially distributed evaluation of occurrences of base metal deposits, however, can be established more precisely in the fuzzy space by assigning membership functions or possibilities by taking "base metal" as a fuzzy restriction and "an object x is a base metal" as the proposition associated with x. Clearly, the final set of resource estimates should be based on the assessment of all exploration data available in the area. Suppose an X-set

$$X = \{D_1, D_2, \ldots, D_n\}$$

is established in the area Ψ, where D_i $(i = 1, 2, \ldots, n)$ is a D-set. The procedure of generating a set that delineates the base metal deposits $T_B = \{t_1, t_2, \ldots, t_m\}$ can

then be sketched as

$$\left.\begin{array}{l} D_1 \\ D_2 \\ \vdots \\ D_n \end{array}\right\} \xrightarrow{G} T_B,$$

where G is an $X - F$ mapping. It can also be expressed as

$$T_B = \{(t_i, F(D_1^i, D_2^i, \ldots, D_n^i))\}$$

where $i = 1, 2, \ldots, m$ and $D_1^i \subseteq D_1, D_2^i \subseteq D_2, \ldots, D_n^i \subseteq D_n$.

The only difference in this procedure from the conventional intuitive approaches is in the way the final results are represented. There is no mathematical analytical relation in the $X - F$ mapping and it is still a very difficult task, especially, when the number of D-sets are large in the area. This problem, however, can be decomposed into subproblems in fuzzy space. The D-sets in the X-set can be individually interpreted and evaluated for an target resource type being searched, and then be individually mapped into the fuzzy space. The fuzzy sets, $F_1, F_2, \ldots, F_n$ obtained from D-sets, can then be combined with fuzzy logic operators in fuzzy space to generate a new fuzzy set that consists of the final objectives. This can be sketched as

$$\left.\begin{array}{l} D_1 \xrightarrow{X-F} F_1 \\ D_2 \xrightarrow{X-F} F_2 \\ \vdots \\ D_n \xrightarrow{X-F} F_n \end{array}\right\} \xrightarrow{FO} T_B \qquad (4.2)$$

where FO is fuzzy operation which combines $F_1, F_2, \ldots, F_n$ to form a new fuzzy set of the resource type (target) being searched.

4.3 Fuzzy Integration

Imprecise and incomplete information, represented using fuzzy sets, can also be processed using fuzzy logic operators. Some of the basic operations which are most frequently applied to information processing are as follows:

(1) Min-operator

The membership function $\mu_F(x)$ of an intersection $F = F_1 \cap F_2 \ldots \cap F_n$ is defined by

$$\mu_F(x) = min\{\mu_1(x), \mu_2(x), \ldots, \mu_n(x)\}. \tag{4.3}$$

This operation is interpreted as logical "AND". If we define a pairwise function

$$g(\mu_1, \mu_2) = min\{\mu_1, \mu_2\}$$

then

$$\begin{aligned} g(g(\mu_1, \mu_2), \mu_3) &= min\{min\{\mu_1, \mu_2\}, \mu_3\} \\ &= min\{\mu_1, min\{\mu_2, \mu_3\}\} \\ &= g(\mu_1, g(\mu_2, \mu_3)). \end{aligned}$$

So, this operator is pairwise and symmetrical.

(2) Max-operator

The membership function $\mu_F(x)$ of a union $F = F_1 \cup F_2 \ldots \cup F_n$ is defined by

$$\mu_F(x) = max\{\mu_1(x), \mu_2(x), \ldots, \mu_n(x)\}. \tag{4.4}$$

This operation is interpreted as logical "OR". Similarly, it is pairwise and symmetrical. The pairwise function is

$$g(\mu_1, \mu_2) = max\{\mu_1, \mu_2\}.$$

(3) Algebraic product operator

The membership function $\mu_F(x)$ of a product $F = F_1 F_2 \ldots F_n$ is defined as

$$\mu_F(x) = \mu_1(x)\mu_2(x)\ldots\mu_n(x). \tag{4.5}$$

This operation can also be interpreted as logical "AND". The pairwise function can be

$$g(\mu_1, \mu_2) = \mu_1\mu_2$$

and obviously it is symmetrical.

(4) Algebraic sum operator

The algebraic sum $F = F_1 + F_2 + \ldots + F_n$ is defined as

$$F = \{(x, \mu_F(x))|x \in A\}$$

where

$$\mu_F(x) = 1 - \prod_{i=1}^{n}(1 - \mu_i(x)). \tag{4.6}$$

Given $n = 2$, we have a pairwise function

$$g(\mu_1, \mu_2) = \mu_1 + \mu_2 - \mu_1\mu_2$$

and

$$\begin{aligned} g(g(\mu_1, \mu_2), \mu_3) &= \mu_1 + \mu_2 + \mu_3 - \mu_1\mu_2 - \mu_2\mu_3 - \mu_3\mu_1 + \mu_1\mu_2\mu_3 \\ &= g(\mu_1, g(\mu_2, \mu_3)). \end{aligned}$$

So, this operator is pairwise and symmetrical. The min-operator does not allow compensation between low and high degrees of membership functions. Full compensation

is assumed by max-operator (Zimmermann, 1985). The algebraic sum operation may also be interpreted as logical "OR", but it not only assumes full compensation but is also increasing. The membership function increases whenever it is combined with a non-zero membership.

(5). γ-operator

The γ-operator was proposed by Zimmermann and Zysno (1980) as a combination of algebraic product and algebraic sum operators. The membership function $\mu_F(x)$ for a γ-aggregation of fuzzy sets $F_1, F_2, \ldots, F_m$ is defined as

$$\mu_F(x) = \left(\prod_{i=1}^{m} \mu_i(x)\right)^{(1-\gamma)} \left(1 - \prod_{i=1}^{m}(1 - \mu_i(x))\right)^{\gamma} \tag{4.7}$$

$$(x \in X,\ 0 \leq \gamma \leq 1).$$

This operator becomes an algebraic product operator (Eq. 4.5) when $\gamma = 0$ and an algebraic sum operator (Eq. (4.6)) when $\gamma = 1$. In these two special cases, it is pairwise and symmetrical. It does not exhibit clear pairwise or symmetry properties when $\gamma \neq 0$ and $\gamma \neq 1$. This means that when $\gamma \neq 1$ and $\gamma \neq 0$, γ-operator does not allow partial possibilities and the membership functions can not be combined hierarchically. The other operators given above allow partial possibilities and the possibilities can be combined one after another. But the γ-operator allows different degrees of compensation depending on the choice of γ. Given a task of combining two or more sets of information, the choice of an appropriate γ value provides a compensatory mechanism in the aggregation of different information categories. Suppose one finds a very high possibility in one data set towards a chosen hypothesis (i.e. exploration target), while another data set fails to show any possibility towards the same hypothesis, or even negative possibility. In such cases, one's total confidence level, estimated by combining two data sets would lie somewhere in between the high

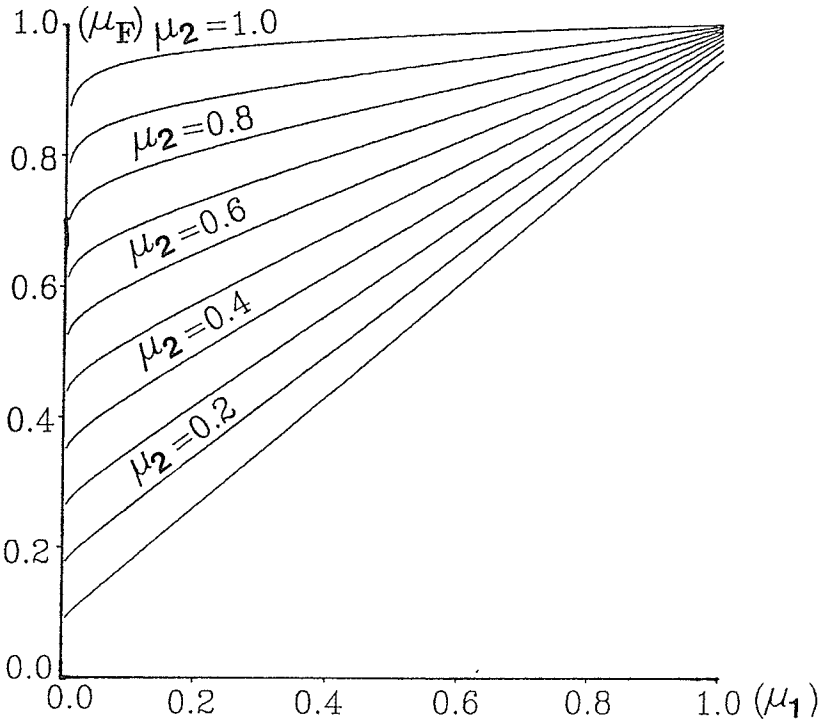

Figure 4.1: Theoretical Property of γ Aggregation ($\gamma = 0.975$)

and low confidence, i.e., the compensation occurs between the high confidence and the low confidence. Degree of compensation between the two extreme confidence levels is determined by the choice of γ. There is no compensation when $\gamma = 0$ and full compensation when $\gamma = 1$. The theoretical curves of the fuzzy set F of aggregating two fuzzy sets F_1 and F_2 using γ-operator with $\gamma = 0.975$ are shown in Figure 4.1. When membership functions are combined using γ-operator that allows a certain level of compensation, a situation exists that the resulting membership function is identical to one of the membership functions. I call the situation, zero aggregation. For example, given two fuzzy sets F_1 and F_2, such that

$$\mu_F = (\mu_1\mu_2)^{1-\gamma}(\mu_1 + \mu_2 - \mu_1\mu_2)^{\gamma}$$

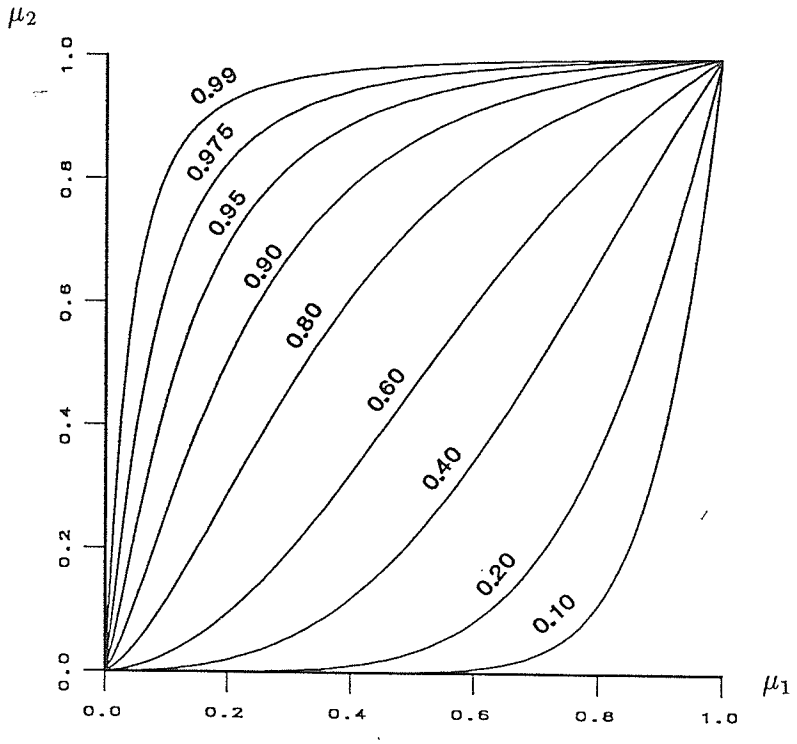

Figure 4.2: Theoretical curves of Zero Aggregation. γ values for each case is listed as shown.

and

$$\mu_F = \mu_1,$$

we have

$$\mu_1 = \frac{\mu_2}{(\mu_2)^{1-\frac{1}{\gamma}} + \mu_2 - 1}.$$

Clearly, zero aggregation depends on the values of μ_1 and μ_2 when a non-zero γ is chosen. The zero aggregation curves with different γ values are plotted in Figure 4.2.

If the combination of two sets is modelled using a *max*-operator, the lower confidence is ignored and one with higher value becomes chosen as the combination of the confidences, as if there does not exist a data set which can actually produce a

low or negative possibility. If the same problem is integrated using an algebraic sum, the total confidence level increases regardless of low or negative confidence values. It is similar with a *min*-operator and an algebraic product operator. Several criteria for selecting appropriate aggregation operators are given by Zimmermann (1985) for choosing optimum operators for a specific model or situation.

4.4 Test Examples

Problems in integrated mineral exploration program can be decomposed into sub-problems based on their specific functions. The exploration target is usually the "top hypothesis" or "proposition". All D-sets in a structured database X-set can be interpreted and evaluated independently for the exploration target and mapped into the fuzzy space. These fuzzy sets are then combined in the fuzzy space to form a fuzzy set of the top proposition. This procedure is sketched in a mathematical notation (Eq. 4.2).

The fuzzy logic method discussed above of integrating geological and geophysical information is tested with the data sets collected from the Farley Lake and Snow Lake areas of northern Manitoba, Canada (Figure 2.1 and 2.2). The top hypotheses or propositions chosen for the Farley Lake area are "there is a base metal deposit" and "there is an iron formation deposit". For Snow Lake area, the hypothesis chosen is "there is a base metal deposit". The fuzzy restrictions for fuzzy subsets, $F_1, F_2, \ldots, F_n$ are "data showing signs of a base metal deposit" and "data showing signs of an iron formation deposit". For example, the fuzzy sets from aeromagnetic data are "magnetic anomalies showing a base metal deposit" and "magnetic anomalies showing an iron formation deposit", etc.

4.4.1 Farley Lake Test Area

There are nine geological and geophysical D-sets available in this area (Figures 2.4-2.12). They are the Bedrock Geology Map, Airborne Magnetic Total Field Map, Ground Electromagnetic Map, VLF Electromagnetic Contour Map (Station: NNS), VLF Electromagnetic Contour Map (Station: NLK), IP Chargeability Contour Map, Ground Resistivity Contour Map, Airborne INPUT Electromagnetic Map, Airborne Electromagnetic Map. Nine fuzzy sets are obtained by assigning membership functions to each data set for each chosen proposition, according to the mineral deposit theory and based on the normal interpretation process of each geophysical data set. For examples, the mafic intrusive rocks and mafic to felsic volcanic rocks should be assigned relatively hihg membership values for the base metal deposit proposition; the iron formation is often associated with strong aeromagnetic anomalies; the higher EM anomaly is more possibly related to a base metal deposit than a lower EM anomaly; an anomaly from a larger scale EM anomaly map can be more effective than an anomaly of similar amplitude from a smaller scale EM anomaly map. The membership functions for each exploration data applied in the subsequent integration experiment are summarized in Table (4.2).

These fuzzy sets are then aggregated by using both the algebraic sum operator (Eq. 4.6) and γ-operator (Eq. 4.7) with $\gamma = 0.975$. As explained in the previous section, choice of $\gamma = 0.975$ means that a certain degree of compensation is allowed in this operation.

The results of combining all membership functions of the fuzzy sets from different D-sets are membership functions of our target hypotheses or propositions, "there is a base metal deposit" and/or "there is an iron formation". The resulting membership functions give us the spatial possibility distribution for each chosen top proposition. They are plotted as grey level maps (Figures 4.3-4.6). Figures 4.3 and 4.5 are com-

Exploration Data (type and value)	$\mu_I(x)$	$\mu_B(x)$
Aeromagnetic Data (γ)		
> 3000	0.25	0.10
500 - 3000	0.15	0.15
< 500	0.10	0.10
Ground EM Data		
< 4	0.05	0.06
4 – 10	0.15	0.14
10 – 20	0.20	0.18
> 20	0.25	0.23
Airborne EM		
B	0.15	0.15
C	0.13	0.13
D	0.11	0.11
E	0.09	0.10
Band	0.07	0.08
No anomaly	0.05	0.05
Ground Resistivity Data (ohm-m)		
< 100	0.25	0.27
100 – 500	0.18	0.20
500 – 1000	0.10	0.13
> 1000	0.05	0.06
IP Chargeability Data (mV/V)		
> 40	0.25	0.27
20 – 40	0.18	0.20
6 – 20	0.10	0.13
< 6	0.05	0.06

Table 4.2: Exploration Data and Membership Functions.

Exploration Data (type and value)	$\mu_I(x)$	$\mu_B(x)$
VLF EM Data (Station: NNs)		
> 80	0.20	0.20
50 – 80	0.15	0.15
20 – 50	0.10	0.10
< 20	0.05	0.06
VLF EM Data (Station: NLK)		
> 80	0.20	0.20
50 – 80	0.15	0.15
20 – 50	0.10	0.10
< 20	0.05	0.06
Airborne INPUT EM Data		
No anomaly	0.08	0.07
Anomaly area	0.10	0.09
Band 2	0.13	0.11
Band 3 and 4	0.16	0.13
Band 5 and 6	0.18	0.15
Bedrock Geological Map		
Felsic Intrusive	0.05	0.05
Basalt – Andesite	0.15	0.15
Iron Rich Rocks	0.25	0.1
Picrite	0.10	0.08
Mafic Intrusives	0.15	0.15

Table 4.2: Exploration Data and Membership Functions (Farley Lake). [$\mu_B(x)$ – membership function for the proposition that there exists a base metal deposit; $\mu_I(x)$ – membership function for the proposition that there exists an iron formation.]

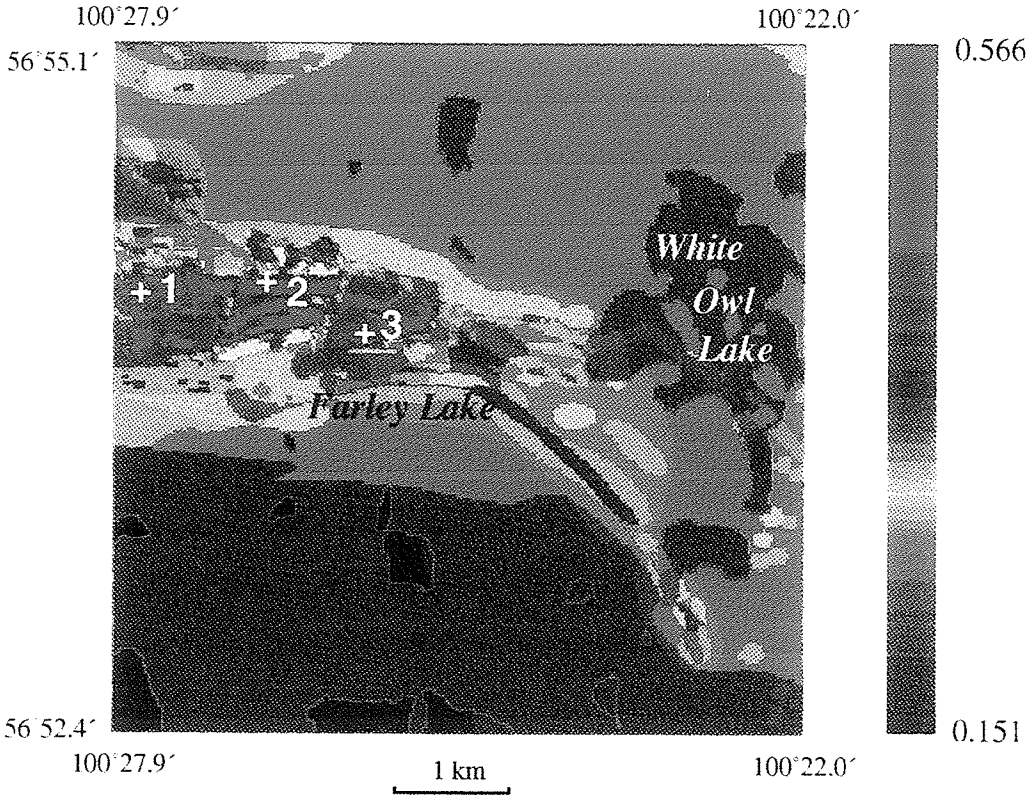

Figure 4.3: Possibility Map for a Base Metal Deposit using γ-Operator (Farley Lake).
1 – Iron; 2, 3 – Gold.

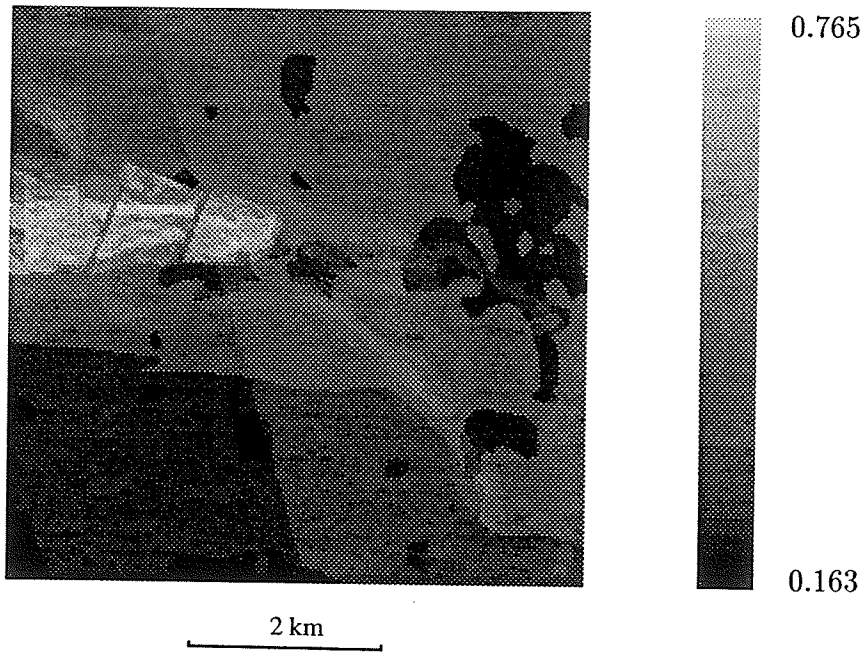

Figure 4.4: Possibility Map for a Base Metal Deposit using Algebraic Sum (Farley Lake).

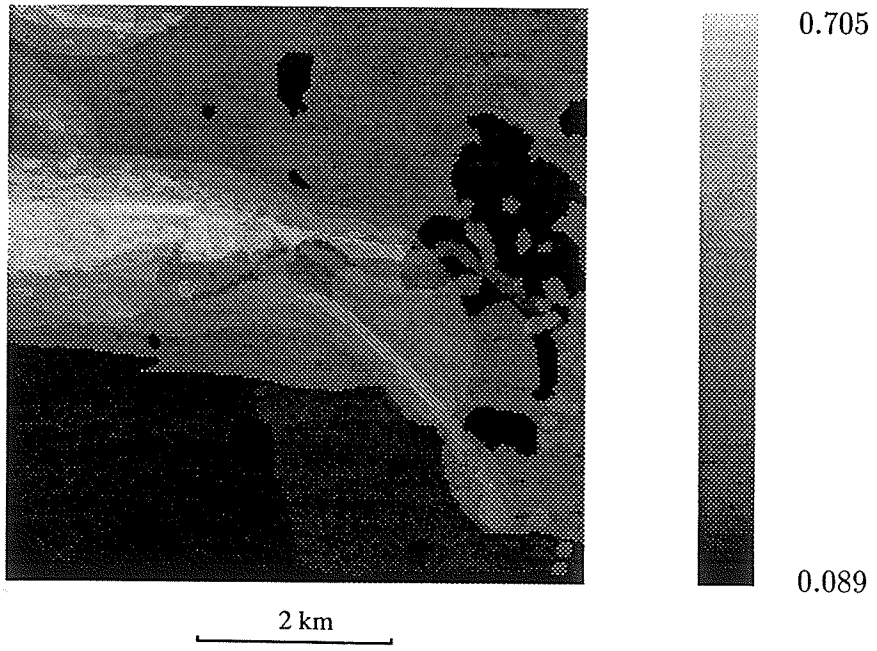

Figure 4.5: Possibility Map for an Iron Formation using γ-Operator (Farley Lake).

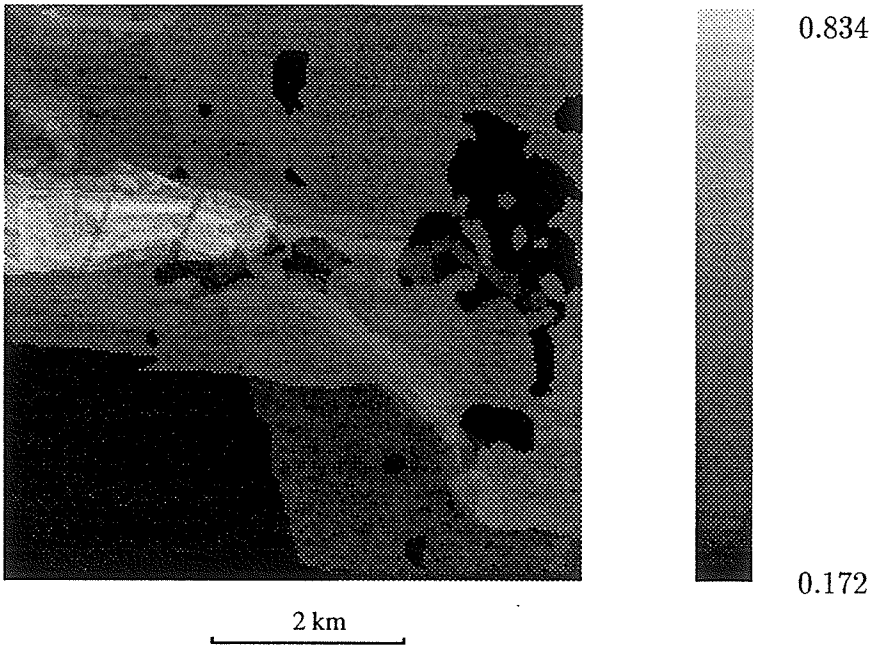

Figure 4.6: Possibility Map for an Iron Formation using Algebraic Sum (Farley Lake).

puted using the γ-operator and Figures 4.4 and 4.6 using the algebraic sum operator. The patterns of the two possibility maps for a base metal deposit, Figures 4.3 and 4.4, are very similar and both have a relatively higher possibility distribution in the west central area. The relative possibility distribution for an iron formation (Figures 4.5 and 4.6) are also similar although the absolute values are quite different.

Many D-sets cover only a small portion of the test area with the remainder having no data (Figures 2.6, 2.7, 2.8, 2.9, 2.10, and 2.12). This fact has significant effect on the choice of γ value. If all data sets had complete coverage, the result would have been expected to have increased resolution when an appropriate γ operator was used. Incomplete spatial coverage of some of these data layers can cause unreasonable results if an improper γ value was chosen. In such cases, it would be safer to use a very high degree of compensation (with γ close to 1) or use an algebraic sum operator.

Values of membership functions are relative for both cases (Table 4.2 and Figures 4.3-4.6). Although the membership is defined in an interval of $[0, 1]$, a membership close to 1 does not necessarily mean near certainty of a mineral deposit. In this study, variation of the originally assigned membership functions by up to $\pm 20\%$ has not significantly altered the pattern of the final possibility maps. This indicates the robustness of the fuzzy set approach.

4.4.2 Snow Lake Test Area

The geological and geophysical data sets available for this test area are Geology Map, Airborne Magnetic Total Field Map, Airborne Magnetic Total Intensity Map, Ground Electromagnetic Anomaly Map, Airborne Electromagnetic Survey Map, Ground Magnetic Survey Map. Among the six D-sets available in this area, two D-sets are both from airborne magnetic sensors. Only one of them is used in this test. As for the Farley Lake area, five fuzzy subsets are obtained by analyzing and evaluating the D-sets independently. The membership functions used to map the D-sets into fuzzy subsets are listed in Table 4.3. The fuzzy subsets are combined by using both the algebraic sum (Eq. 4.6) and γ-operator (Eq. 4.7) with $\gamma = 0.975$.

The final result is the fuzzy set with the proposition that "there exists a base metal deposit" from all five D-sets and the final membership functions represent the spatial distribution of possibilities that there is a base metal deposit. The possibility maps (Figures 4.7 and 4.8) obtained using different operators appear the same. Both of them have characteristically high possibilities in the east central part of the area.

Exploration Data (type and value)	$\mu_B(x)$
Airborne EM Data	
Very strong anomaly	0.12
Strong anomaly	0.10
Medium Anomaly	0.08
Week anomaly	0.05
No anomaly	0.01
Ground EM Data	
Strong anomaly	0.2
Medium anomaly	0.15
Week anomaly	0.10
No anomaly	0.01
Ground Magnetic Data	
Strong anomaly	0.10
Week anomaly	0.07
No anomaly	0.01
Geological Map	
Felsic intrusions	0.01
Mafic intrusions	0.07
Quartzofeldspathic gneiss and migmatite	0.01
Sandstone and conglomerate	0.05
Greywacke, siltstone and mudstone	0.08
Felsic volcanic rocks	0.10
Mafic to intermediate volcanic rocks	0.10
Airborne Magnetic Data γ	
> 3000	0.08
2550 – 3000	0.10
< 2550	0.01

Table 4.3: The Available Exploration D-sets and Membership Functions (Snow Lake). $\mu_B(x)$ – membership function for the proposition that there exists a base metal deposit.

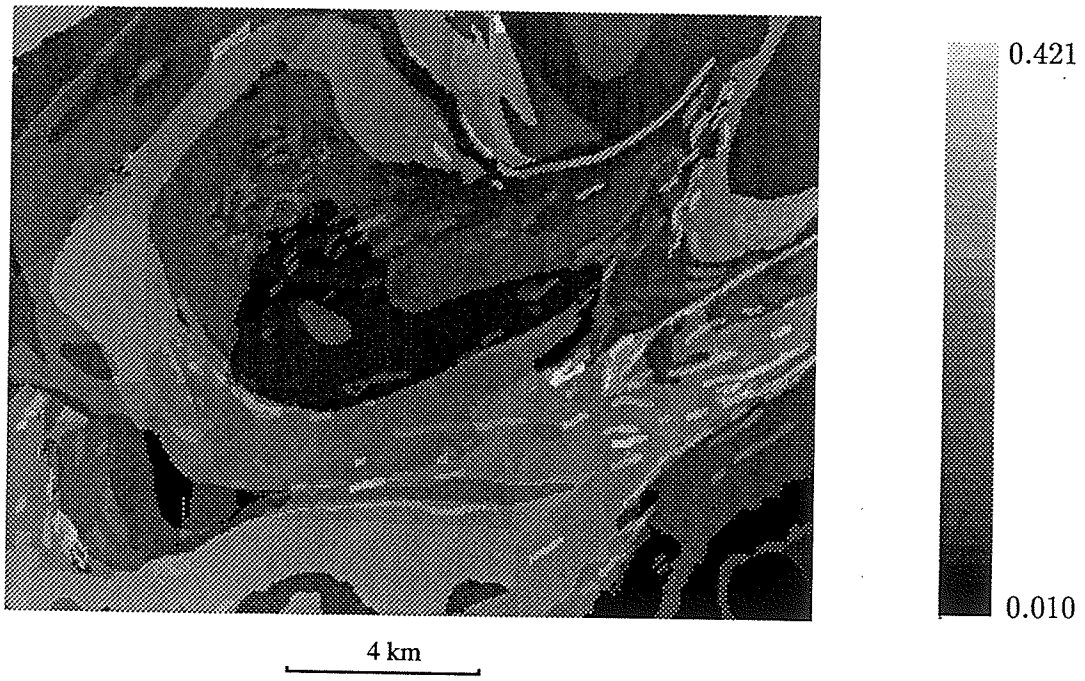

Figure 4.7: Possibility Map for a Base Metal Deposit using γ-operator (Snow Lake).

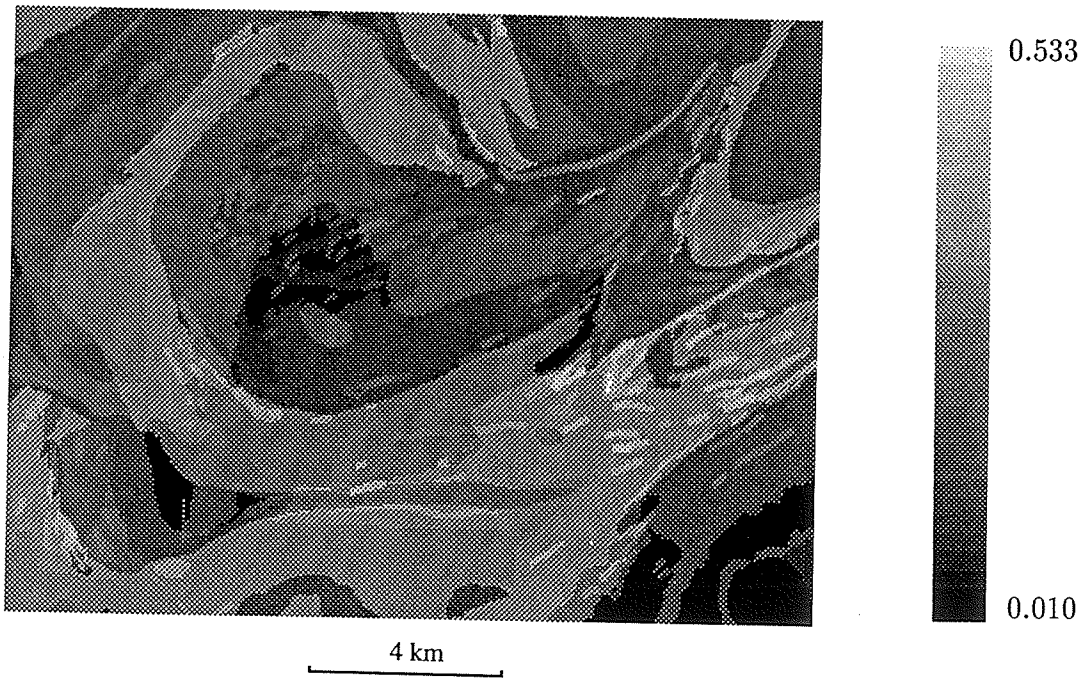

Figure 4.8: Possibility Map for a Base Metal Deposit using Algebraic Sum (Snow Lake).

4.5 Jackknife Analysis

Combining the membership functions from different D-sets using the algebraic sum or γ-operator results in an estimate of the unknown membership function associated with a specific target proposition. As the number of D-sets increases, it becomes important to estimate the relative contribution and/or bias of one D-set with respect to others. The jackknife technique can be used to estimate the bias in the membership estimate and the jackknife variance can be computed to evaluate the estimate.

4.5.1 Jackknife Technique

Let θ be an unknown membership function and $\mu_1^i, \mu_2^i, \ldots, \mu_n^i$ be a sample of size n of independent and identically distributed fuzzy membership functions from different D-sets. Suppose $\widehat{\theta}$ is an estimate of the unknown membership function θ and $\widehat{\theta}$ is obtained by combining n layers of D-sets using a specific fuzzy operator. Let $\widehat{\theta}_{-i}$ be the corresponding estimate obtained using the same fuzzy operator based on the sample of size $(n-1)$, where the ith fuzzy membership function has been deleted. Then, one can calculate

$$\widetilde{\theta}_i = n\widehat{\theta} - (n-1)\widehat{\theta}_{-i} \tag{4.8}$$

where $i = 1, \ldots, n$.

The quantity

$$\widetilde{\theta} = \frac{1}{n}\sum_{i=1}^{n}\widetilde{\theta}_i \tag{4.9}$$

has the property that it eliminates the first order bias from $\widehat{\theta}$. Namely, if

$$E(\widehat{\theta}) = \theta + c_1/n + O(1/n^2), \tag{4.10}$$

where c_1/n is the first order bias and $O(1/n^2)$ includes the terms of second and higher

order bias, then

$$E(\tilde{\theta}) = \theta + O(1/n^2). \tag{4.11}$$

From (Eq. 4.10 and 4.11), we have the first order bias

$$c_1/n = E(\hat{\theta}) - E(\tilde{\theta}) \tag{4.12}$$

and its absolute value

$$|c_1/n| = |E(\hat{\theta}) - E(\tilde{\theta})| \tag{4.13}$$

is, in practice, more convenient to use. In an unpublished work, Tukey created the name "jackknifed estimater" for (Eq. 4.9) in the hope that it would be a rough-and-ready statistical tool (Miller, 1974) and $\tilde{\theta}$ is called jackknife estimation.

Tukey (1958) proposed that $\tilde{\theta}_i$ $(i = 1, \ldots, n)$ could, to a close approximation, be treated as n independent estimates and it is in many instances identically distributed when n is large. Then,

$$\frac{1}{n(n-1)} \sum_{i=1}^{n} (\tilde{\theta}_i - \tilde{\theta})^2 \tag{4.14}$$

becomes an estimate of jackknife variance $Var(\tilde{\theta})$ (Miller, 1968).

As shown above, the jackknife technique can be used to estimate the first order bias. In this research, it is also used to evaluate the consistent validity of the fuzzy operators used to combine the membership functions. If the resulting pattern of the spatial distribution of $\mu_F(x) = E(\hat{\theta})$ is similar to the spatial distribution pattern of the jackknife estimate $E(\tilde{\theta})$, the fuzzy operator used is judged acceptable in this research. If they are substantially different, the operator and each D-set have to be re-examined.

The jackknife estimater (Eq. 4.9) does not confine its estimations within $[0, 1]$ although the membership, as shown above, is defined in $[0, 1]$. Because it only has relative meaning and can easily be normalized without altering the original informa-

tion content, it is not a serious problem that the jackknife estimation is in some cases larger than 1.

4.5.2 Farley Lake Area

The jackknife estimation, jackknife variance, and first order bias of the possibilities distribution obtained using both the algebraic sum and γ-operator are computed for the two exploration target propositions. Because the conclusions from both propositions are very similar, only the results for the proposition that there exists a base metal deposit are discussed below.

Comparing the relative spatial possibility distributions shown in Figures 4.3 and 4.4 with their corresponding jackknife estimations (Figures 4.9, 4.10), there are no significant difference between them.

The jackknife variances are computed using (Eq. 4.14) and shown in Figures 4.11 and 4.12. The two jackknife variance maps obtained using algebraic sum and γ-operator have very similar spatial distribution patterns. The high variance values are found in some parts of the west central test areas, where some D-sets give high membership functions towards the target proposition while the others can only give very low membership functions towards the same proposition. The spatial distribution is clearly affected by the incomplete spatial coverage of some D-sets. The low variance values in the southwest and lake areas are due to the lack of available spatial data layers.

The results of the first order bias $|c_1/n|$ of the possibility maps for the proposition that there exists a base metal deposit using both operators are shown in Figures 4.13 and 4.14. In the west central test area where there are more spatial data layers, Figure 4.13 shows a relatively low first order bias, whereas Figure 4.14 shows a relatively high first order bias. Further more, the first order bias ranges between 0.051 and 0.186 with

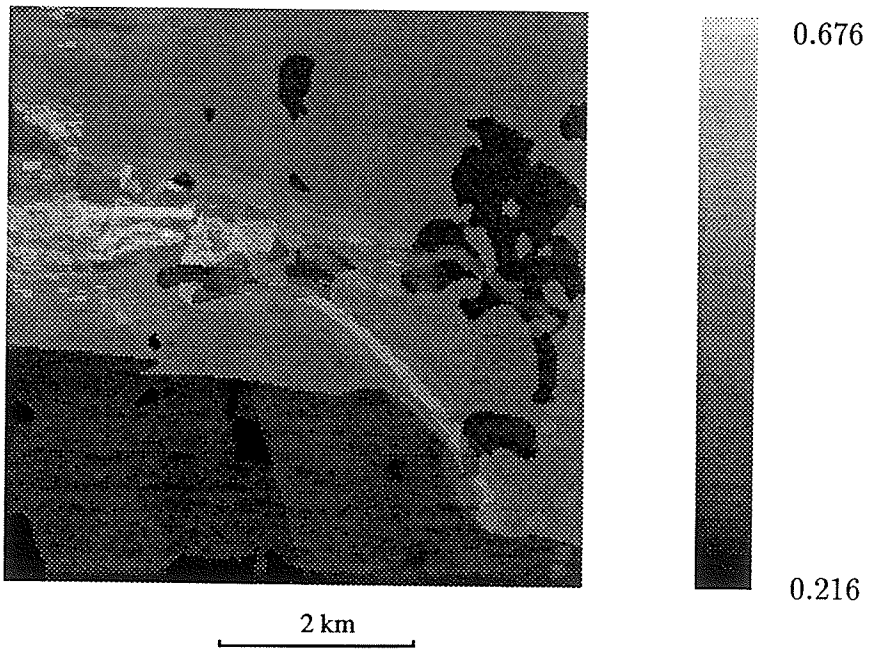

Figure 4.9: Jackknife Estimation of Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake).

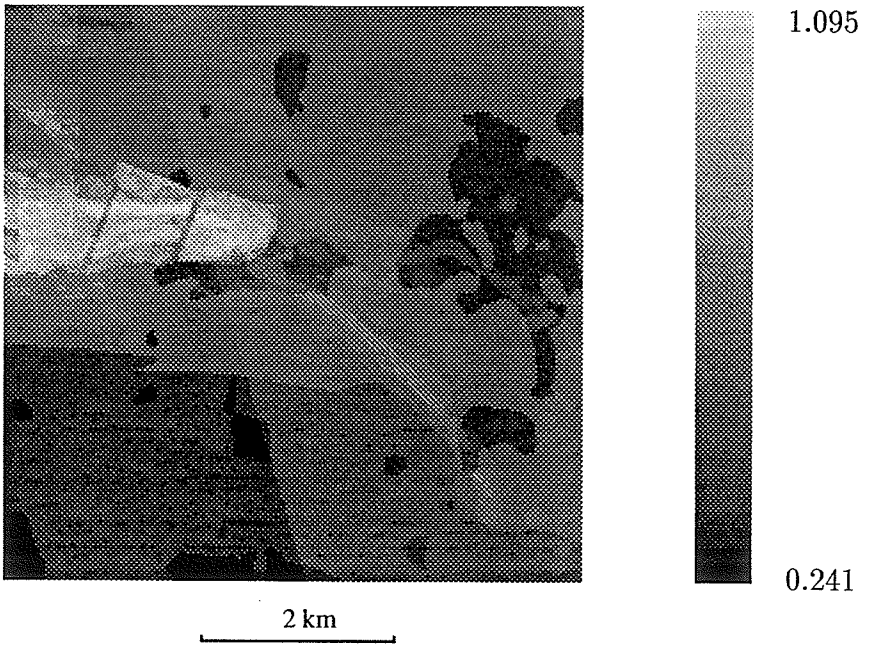

Figure 4.10: Jackknife Estimation of Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake).

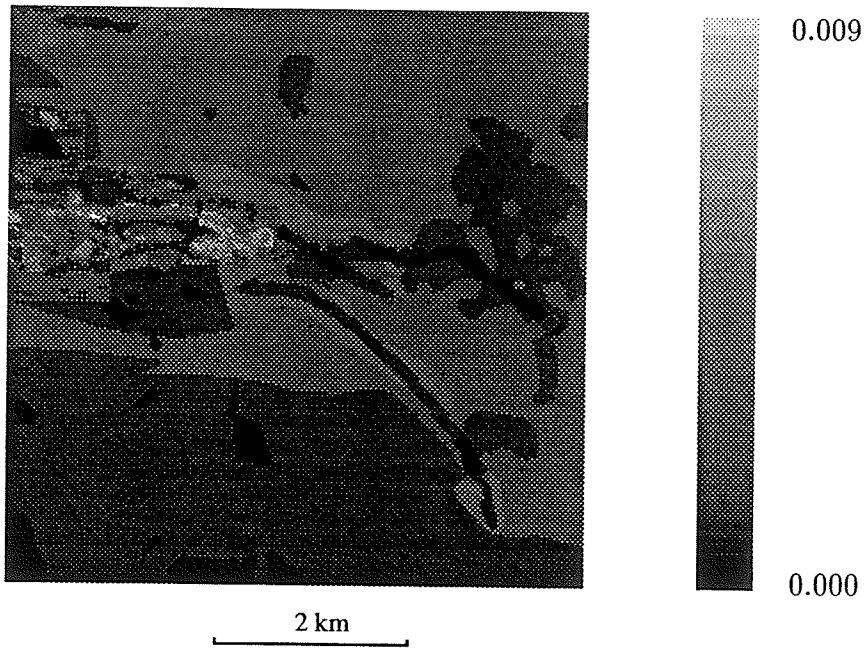

Figure 4.11: Jackknife Variance of Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake).

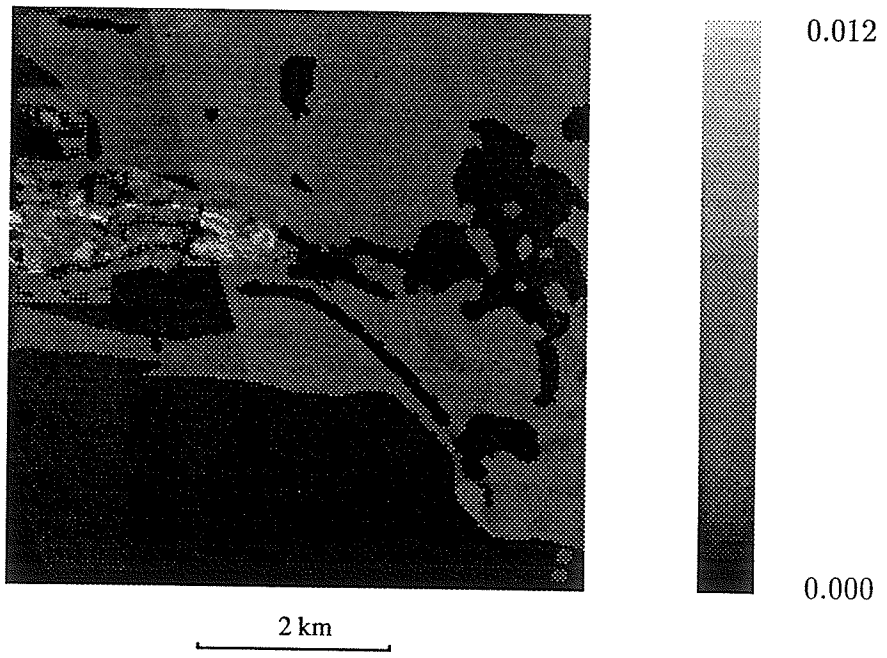

Figure 4.12: Jackknife Variance of Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake).

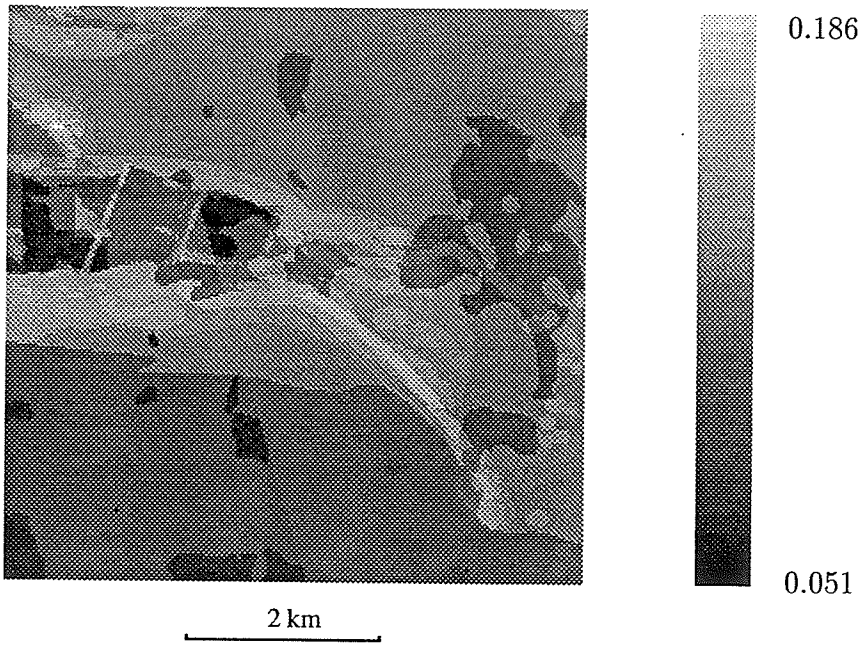

Figure 4.13: First Order Bias in the Possibilities for a Base Metal Deposit using γ-Operator (Farley Lake).

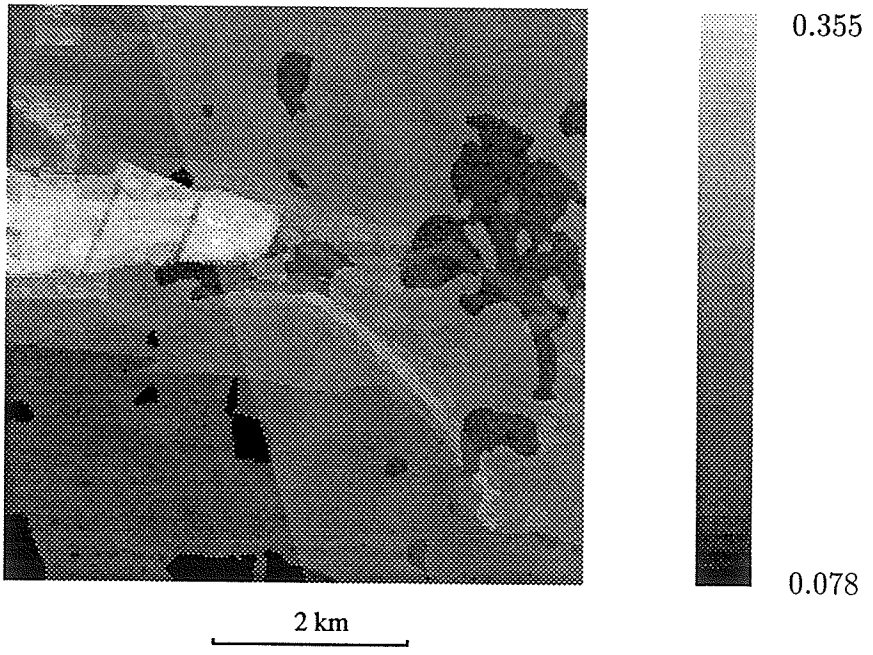

Figure 4.14: First Order Bias in the Possibilities for a Base Metal Deposit using Algebraic Sum (Farley Lake).

the γ-operator, and from 0.078 to 0.355 with the algebraic sum operator. The results obtained using γ-operator appear to be less biased than the case with algebraic sum. These facts indicate that the first order bias is related to the combination operators. The γ-operator allows a certain degree of compensation while the algebraic sum is increasing. Another important factor that affects the extent of the first order bias is the spatial coverage of the D-sets. This is shown by the fact that the value of the first order bias is in general relatively low in the lake areas.

4.5.3 Snow Lake Area

The results of the jackknife estimation of the possibilities for the proposition that there exists a base metal deposit using the algebraic sum and γ-operator are shown in Figures 4.15 and 4.16. As in the Farley Lake test area, their patterns of spatial distribution are very similar to their corresponding spatial possibility distributions (Figures 4.7 and 4.8). The corresponding jackknife variance distributions are shown in Figures 4.17 and 4.18. Again similar to the Farley Lake test area, high variances are associated with areas where contradictory fuzzy membership functions are obtained from different D-sets. Clearly they are affected by the incomplete spatial data coverages. The spatial distribution of the first order bias estimated using the γ-operator and algebraic sum are shown in Figures 4.19 and 4.20. Their patterns are similar and the values range from 0.0 to 0.165 with γ-operator and from 0.0 to 0.292 with algebraic sum operator. Similar to the Farley Lake area, it appears that the results are less biased than those obtained using an algebraic sum operator.

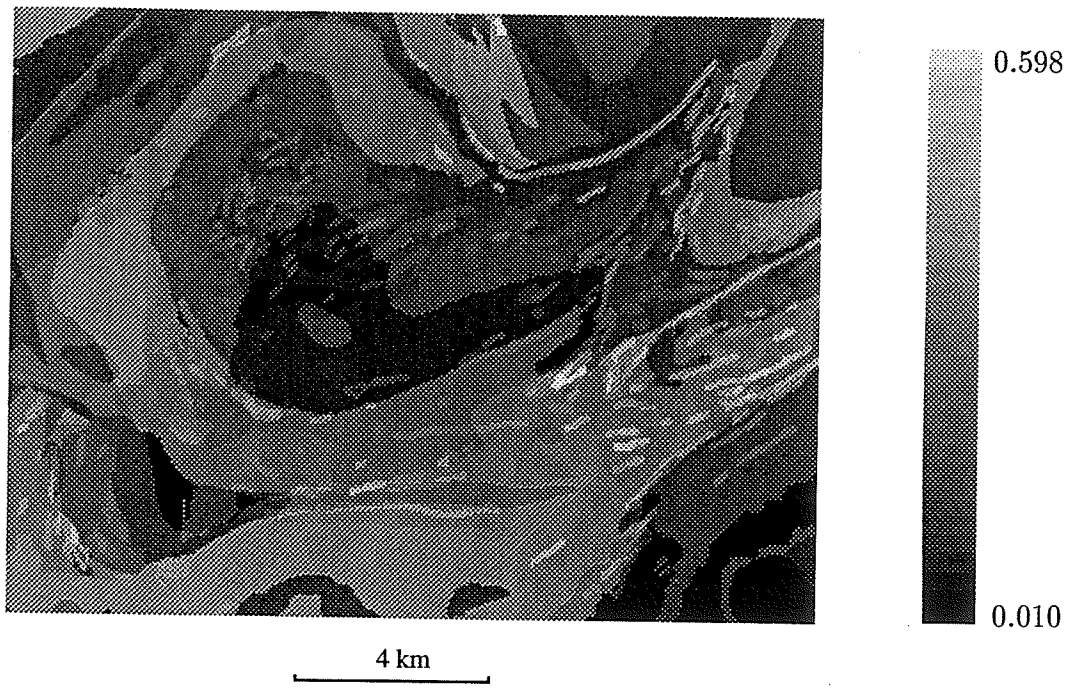

Figure 4.15: Jackknife Estimation of the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake).

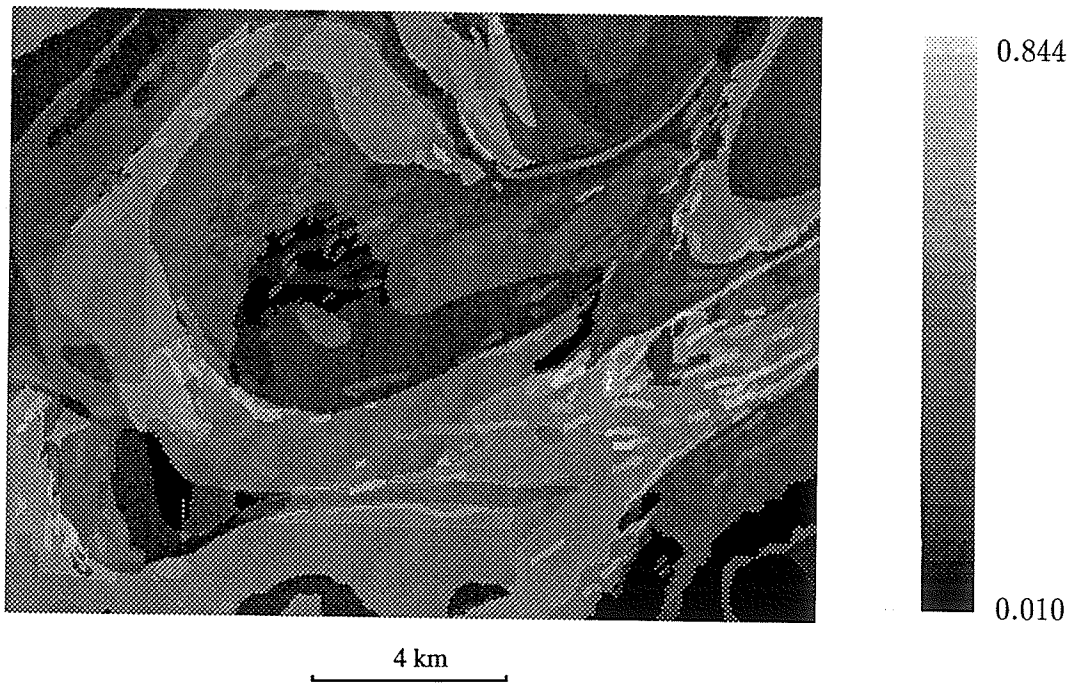

Figure 4.16: Jackknife Estimation of the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake).

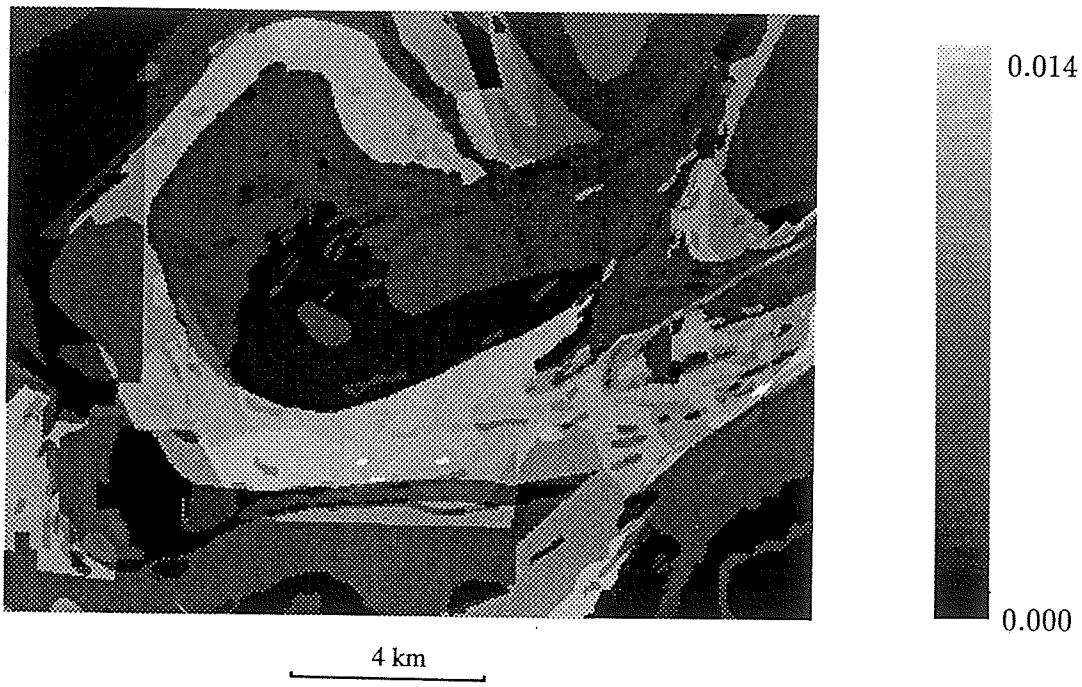

Figure 4.17: Jackknife Variance of the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake).

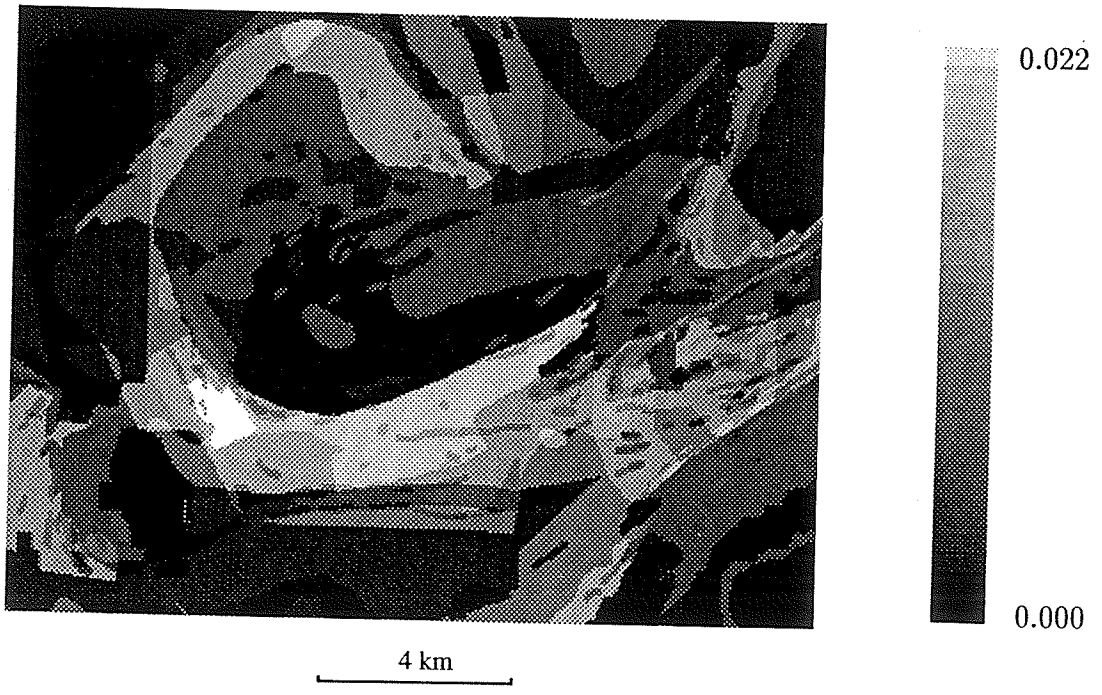

Figure 4.18: Jackknife Variance of the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake).

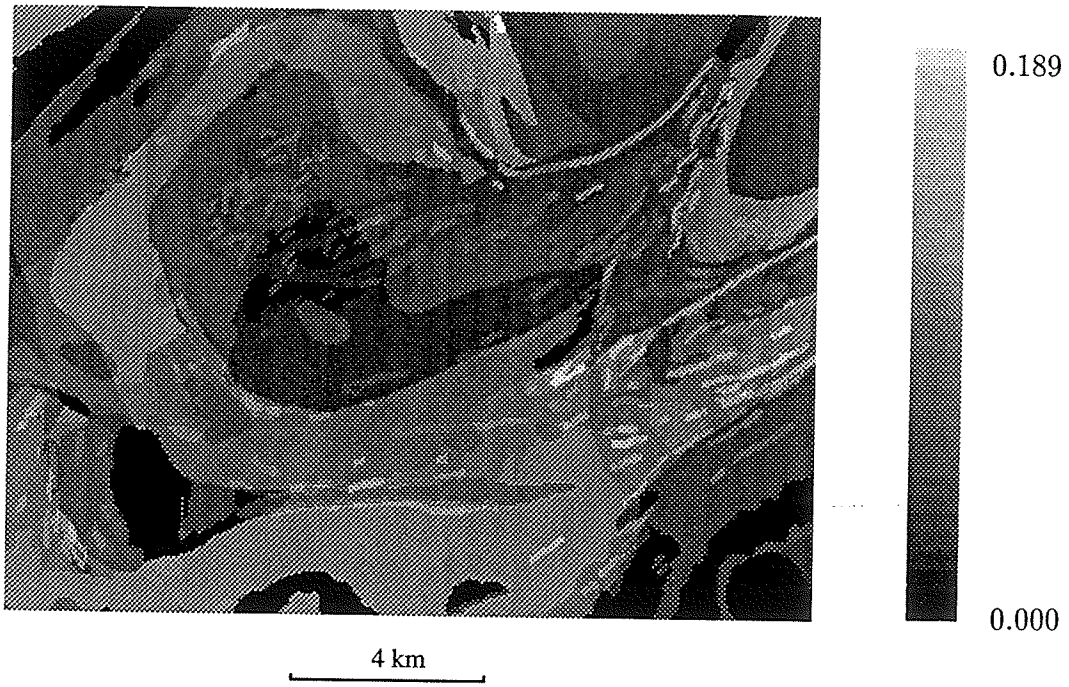

Figure 4.19: First Order Bias in the Possibilities for a Base Metal Deposit using γ-Operator (Snow Lake).

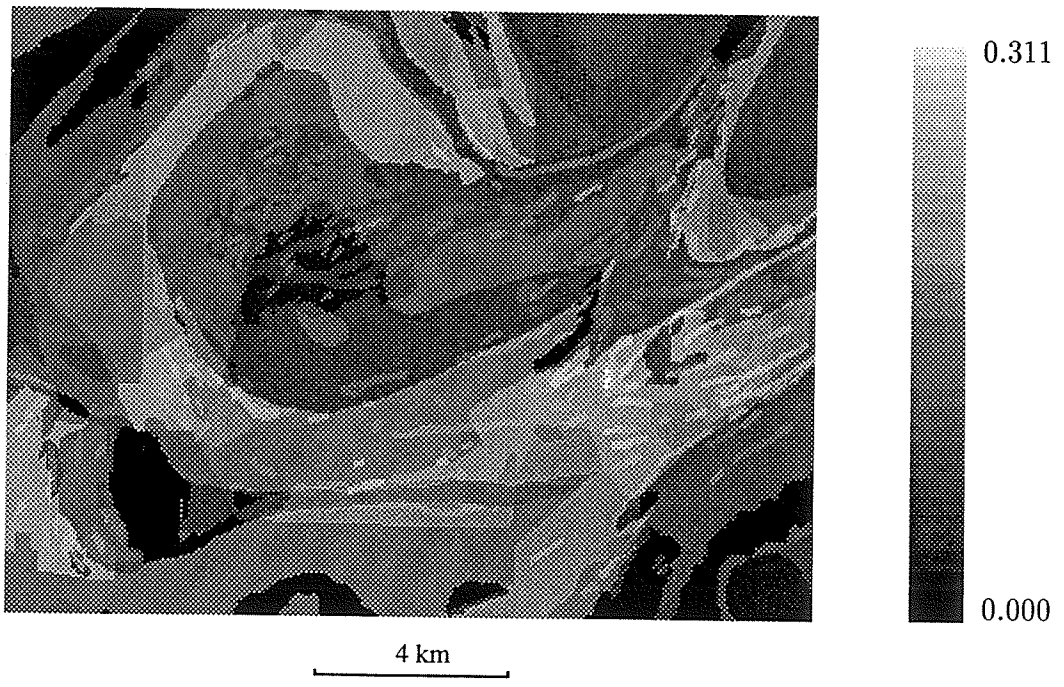

Figure 4.20: First Order Bias in the Possibilities for a Base Metal Deposit using Algebraic Sum (Snow Lake).

4.6 Error analysis

The whole integration procedures discussed in previous sections include independent interpretation of different D-sets and mapping of the data into a fuzzy space by assigning membership functions to each data object. Errors naturally occur and exist from various sources, some of which are random, observation, digitizing, and interpolation errors, noise in the data, interpreter's uncertain knowledge, etc. These errors are reflected in the fuzzy membership functions assigned to different D-sets and subsequently propagate through the entire reasoning process towards the final membership function. In this section, some formulae derived in this research are described for error propagation through the aggregation process and interpretation of relative information is discussed.

4.6.1 Error Propagation through the Aggregation Processes

Let $\widehat{\mu}$ be an estimate of resulting membership function μ and

$$\widehat{\mu} = \mu + \epsilon$$

where ϵ is error term. The algebraic product of n membership functions is given as

$$\mu = \prod_{i=1}^{n} \mu_i.$$

Let μ_{-i} be the algebraic product with the ith term deleted, i.e.

$$\mu_{-i} = \mu_1 \ldots \mu_{i-1} \mu_{i+1} \ldots \mu_n.$$

Then, we have

$$\frac{\partial \mu_p}{\partial \mu_i} = \mu_{-i}.$$

Let ϵ_i be the error term associated with the membership function from the ith D-set. The error, ϵ_p, associated with algebraic product can be estimated using Taylor expansion with the second and higher terms ignored and it is given as

$$\begin{aligned} \epsilon_p &= \sum_{i=1}^{n} \mu_{-i}\epsilon_i \\ &= \mu_p \sum_{i=1}^{n} \frac{\epsilon_i}{\mu_i}. \end{aligned} \tag{4.15}$$

The ϵ_p can then be further normalized using the resulting membership function μ. The normalized error is called relative error. It is also an important value for evaluating the estimated resulting membership function. The relative error ε_p associated with algebraic product is given by

$$\varepsilon_p = \frac{\epsilon_p}{\mu} = \sum_{i=1}^{n} \frac{\epsilon_i}{\mu_i}.$$

If $\frac{\epsilon_i}{\mu_i}$ can be referred as relative error of the membership function from the ith D-set, the relative error associated with algebraic product is the sum of the relative errors from membership functions from different D-sets.

The algebraic sum

$$\begin{aligned} \mu &= 1 - \prod_{i=1}^{n}(1-\mu_i) \\ &= 1 - \nu \end{aligned}$$

where $\nu = \prod_{i=1}^{n}(1-\mu_i(x))$. Now let

$$\nu_{-i} = (1-\mu_1)\ldots(1-\mu_{i-1})(1-\mu_{i+1})\ldots(1-\mu_n).$$

The error, ϵ_s, associated with algebraic sum can be estimated by neglecting the second and higher terms in the Taylor expansion and it is given as

$$\begin{aligned} \epsilon_s &= \sum_{i=1}^{n} \frac{\partial \mu_s}{\partial \mu_i}\epsilon_i \\ &= \sum_{i=1}^{n} \nu_{-i}\epsilon_i. \end{aligned} \tag{4.16}$$

Similarly, the relative error, ε_s, associated with algebraic sum can then be estimated by

$$\varepsilon_s = \frac{\epsilon_s}{\mu} = \frac{\nu}{1-\nu} \sum_{i=1}^{n} \frac{\epsilon_i}{1-\mu_i}.$$

In the case of the γ-operator, we have

$$\mu = \left(\prod_{i=1}^{n} \mu_i \right)^{(1-\gamma)} \left(1 - \prod_{i=1}^{n} (1-\mu_i) \right)^{\gamma}$$

$$(x \in X,\ 0 \leq \gamma \leq 1).$$

Similarly as for the algebraic sum, the error, ϵ, associated with γ-operator can be estimated as

$$\epsilon_r = \mu \sum_{i=1}^{n} \left[(1-\gamma) \frac{\epsilon_i}{\mu_i} + \gamma \frac{\nu_{-i}\epsilon_i}{1-\nu} \right] \tag{4.17}$$

$$(x \in X,\ 0 \leq \gamma \leq 1).$$

Note that the above equation becomes (Eq. 4.16) when $\gamma = 1$ and (Eq. 4.15) when $\gamma = 0$. The corresponding relative error ε_r can be defined as

$$\varepsilon_r = \frac{\epsilon_r}{\mu} = (1-\gamma)\varepsilon_p + \gamma\varepsilon_s.$$

Using the above formulae to estimate the errors passing through the aggregation process requires quantitative estimation of errors in the data. Because this information is not available, these formulae are not used in the real exploration data processing in this thesis research.

4.6.2 Interpretation of Relative Information

As mentioned above, the membership functions are relative. A membership close to 1 does not necessarily mean near certainty towards the target proposition. If the

membership value is higher at a location Ψ_1 than at another location Ψ_2, however, it does mean that the the possibility for the target proposition is higher at the location Ψ_1 than at the location Ψ_2 based on the information from the whole X-set.

The algebraic sum and γ-operator are also proportional. A higher membership function can be expected if a membership function is combined with another larger membership function. This property of the combination rules allows a relative $X - F$ mapping and the membership functions can be very distinct from their real values. Actually, we do not know exactly what their membership values are, but in most cases we can reason and learn which data values should have a higher membership function and which D-set is more effective for the target. For example, (1) the high anomaly areas in a D-set representing ground EM data objects should have higher membership for a base metal deposit proposition and the areas with no anomaly should have lower membership, and (2) if there are two D-sets, one is ground EM and the other is airborne magnetic, clearly the anomalies in the D-set of EM data objects should have higher membership for a base metal deposit proposition than the anomalies in the D-set of airborne magnetic data objects. At this point two types of important relationships in the $X - F$ mapping should be pointed out: the relations inside each D-set and the relations between different D-sets. If these two distinct types of relations can be established without conflict in an exploration area, resulting high level membership functions usually have higher possibilities for the target objects being searched.

4.7 Results and Discussion

Inadequacy of the conventional classical set theoretical approach of combining geological and geophysical exploration data has been known for some time. There is an urgent need for developing and establishing mathematical theories to precisely

represent and integrate spatially interpolated geological and ground and/or airborne geophysical data. In this chapter, a mathematical basis for using the fuzzy logic theory is developed and illustrated with real exploration data from the mineral exploration projects in the northern Manitoba, Canada.

The results indicate that the fuzzy set theory provides an effective tool that can adequately represent and manage the imprecise and incomplete information contained in each of the geological and geophysical data sets. The possibility maps for each of the top hypotheses "there is a base metal deposit" and "there is an iron formation deposit" in the Farley Lake area and Snow Lake area do outline the most favorable areas, much more accurately and efficiently than the conventional intuitive integration approaches. In fact, mineral deposits such as sulphide facies iron formation and gold deposits were discovered in the areas outlined by the study in the Farley Lake area (Chevillard and Genaile, 1970).

There is no significant difference between the possibility distribution patterns from the simple combination and the their corresponding jackknife estimation. It appears that the results obtained using γ-operator are less biased than using algebraic sum operator.

Incomplete coverage of some of spatial data layers can cause unreasonable results when an improper γ value is chosen. In such cases, a very high compensation (γ near 1) or an algebraic sum operation is recommended. If all data sets have complete coverage of the exploration area, less biased final possibility distribution can be expected when the γ-operator is used with an appropriate γ value.

In conclusion, the test results strongly support effectiveness of the fuzzy logic approach of integrating multiple spatial geological and geophysical D-sets. One of the deficiencies of this method, at present, appears to be that there is no adequate way to distinguish between negative evidence and lack of evidence as in the Dempster-Shafer

evidential belief function approach (see Chapter 5). Low possibility values computed in the final results cannot be properly interpreted because they may either due to lack of data or due to very low membership function values. Another difficulty in applying fuzzy logic approach is that there is presently no standard aggregation operator, the fact of which, in certain situations, provides flexibility but possibly causes confusion and non-uniqueness to the solution.

Chapter 5

Evidential Belief Function and Integrating Geological and Geophysical Data Sets

Evidential belief function theory is based primarily on Dempster's work on generalization of Bayesian inference and upper and lower probabilities (Dempster, 1966, 1967a, 1967b, 1968). In the belief function theory, Shafer (1976) reinterpreted the lower probabilities as epistemic probabilities or degree of belief and upper probabilities as plausibilities. In this chapter, I will interpret belief functions as probabilities in the traditional probabilistic framework.

In recent years, the belief function theory has received considerable attention in the field of AI (Lee, 1987) and geoscientists have tried to apply it in both development of expert systems and in GIS related geo-information integration (An, 1989; Moon, 1990; Kim and Swain, 1989). Belief function theory has the following advantages over the Bayesian probability theory:

- It subsumes the Bayesian probability theory. The Bayesian probability theory is a limited case of the belief function theory.

- It is able to represent ignorance. The Bayesian probability does not discriminate between lack of belief and disbelief. When one commits a degree of belief to a proposition, the remainder is automatically committed to its negation. Belief function theory, however, does not require the whole remainder is committed to its negation and the degree of disbelief can be determined based on one's knowledge or on statistical analysis. We will see later in this chapter that this characteristic of the belief function theory provides a more adequate basis to create a realistic picture about the corresponding proposition.

- It allows beliefs from several sources to be treated symmetrically and pooled together. In the framework of belief function theory, combination of evidence is achieved using the Dempster's rule of combination. Because the Dempster's rule is symmetrical, the evidence from different sources can be combined one after another without concern about which evidence is old and which evidence is new.

In this chapter, some of the important ideas of belief function theory and Dempster's rule of combination will be briefly reviewed, and then the issues of representation and integration of geoscience information using this theory will be described and tested with real exploration data from both Farley Lake and Snow Lake test areas. Error propagation and conflict information problems will then be discussed.

5.1 Dempster-Shafer Evidential Belief Function Theory

The basic concepts that form the backbone of Shafer's belief function theory are the *frame of discernment, basic probability numbers, belief functions,* and *plausibility functions.* A frame of discernment is a set of all different possibilities under

consideration (Shafer, 1976). It is denoted by Θ. The propositions of interest are in a one-to-one correspondance with the subsets of Θ. Thus, 2^{Θ} denotes the set of all subsets of Θ and corresponds to the set of all propositions of interest. When a proposition corresponds to a subset of a frame of discernment, the frame is said to "discern" that proposition.

The definition of a basic probability assignment is as follows: If Θ is a frame of discernment, then a function $m : 2^{\Theta} \rightarrow [0,1]$ is called a *basic probability measure* whenever

$$m(\emptyset) = 0 \tag{5.1}$$

$$\sum_{H \subset \Theta} m(H) = 1, \tag{5.2}$$

where H is a subset of Θ. The quantity $m(H)$ is called H's *basic probability number*, and it is understood to be the measure of the belief that is committed exactly to H. Condition (5.1) reflects the fact that no belief ought to be committed to an empty set $\emptyset$, while condition (5.2) reflects the convention that one's total belief has a measure of one. The quantity $m(H)$ measures the belief that one commits exactly to H, not the total belief that one commits to H. To obtain the total belief committed to H, one must add to $m(H)$ the quantities $m(S)$ for all proper subsets S of H

$$Bel(H) = \sum_{S \subset H} m(S). \tag{5.3}$$

If Θ is a frame of discernment, then a function $Bel : 2^{\Theta} \rightarrow [0,1]$ is a belief function, if and only if it satisfies the following condition:

$$Bel(\emptyset) = 0 \tag{5.4}$$

$$Bel(\Theta) = 1 \tag{5.5}$$

and for every positive integer n and every collection $H_1, \ldots, H_n$ of a subset of Θ

$$Bel(H_1 \bigcup \cdots \bigcup H_n) \geq \sum_{\substack{I \subset \{1,\cdots,n\} \\ I \neq \emptyset}} (-1)^{|I|+1} Bel\Big(\bigcap_{i \in I} H_i\Big). \tag{5.6}$$

Clearly, the Eqs. (5.4) and (5.5) are analogous to Eqs. (5.1) and (5.2). Taking $n = 2$, Eq. (5.6) becomes

$$Bel(H_1 \cup H_2) \geq Bel(H_1) + Bel(H_2) - Bel(H_1 \cap H_2). \tag{5.7}$$

By further assuming $H_1 \cap H_2 = \emptyset$, and taking only the equal sign of Eq. (5.7), we have

$$Bel(H_1 \cup H_2) = Bel(H_1) + Bel(H_2). \tag{5.8}$$

This is the addability condition of the Bayesian probability. The equations (5.8), (5.4), and (5.5) together define a Bayesian probability.

A subset H of a frame Θ is called a *focal element* of a belief function Bel over Θ if $m(H) > 0$. The union of all the focal elements of a belief function is called its *core*. A plausibility function $Pl : 2^\Theta \rightarrow [0, 1]$ is defined using belief function Bel as

$$\begin{aligned} Pl(H) &= 1 - Bel(\overline{H}), \\ &= \sum_{S \subset \Theta} m(S) - \sum_{S \subset \overline{H}} m(S), \\ &= \sum_{S \cap H \neq \emptyset} m(S) \end{aligned} \tag{5.9}$$

for every $H \subset \Theta$, and where $\overline{H}$ is the negation of H. Comparing Eq. (5.9) with Eq. (5.3) we have

$$Bel(H) \leq Pl(H). \tag{5.10}$$

The belief and plausibility about H, $Bel(H)$ and $Pl(H)$ can be viewed as the lower and upper probabilities. The degree of uncertainty about H is represented by

$$Pl(H) - Bel(H) \tag{5.11}$$

and when it equals 0, we have

$$Bel(H) + Bel(\overline{H}) = 1. \tag{5.12}$$

It becomes a Bayesian probability.

5.2 Dempster's Rule of Combination

Dempster's rule of combination (Dempster, 1967; Shafer, 1976) is a mathematical approach to combine two or more belief functions. Suppose m_1 is the basic probability assignment for a belief function Bel_1 over a frame Θ, and its focal elements are $H_1, \ldots, H_h$. Then the probability masses are measured by the basic probability numbers, $m_1(H_1), \cdots, m_1(H_h)$. Suppose there is a second belief function Bel_2 with basic probability assignment m_2, focal elements $S_1, \ldots, S_l$ and the basic probability numbers $m_2(S_1), \cdots, m_2(S_l)$. Suppose

$$k = \sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i) m_2(S_j) < 1. \tag{5.13}$$

Then the function $m : 2^\Theta \rightarrow [0, 1]$ defined by $m(\emptyset) = 0$ and

$$m(H) = \frac{1}{1-k} \sum_{\substack{i,j \\ H_i \cap S_j = H}} m_1(H_i) m_2(S_j) \tag{5.14}$$

for all non-empty $H \subset \Theta$ is a *basic probability number*. The core of the belief function given by m is equal to the intersection of the cores of Bel_1 and Bel_2.

The basic probability for the new belief function $m(H)$ is also called the *orthogonal sum* of Bel_1 and Bel_2. The combination of Bel_2 and Bel_2 can be denoted as

$Bel_1 \oplus Bel_2$. Because Dempster's rule of combination (DRC) is symmetrical, combining a collection of belief functions can be performed by

$$(\ldots((Bel_1 \oplus Bel_2) \oplus Bel_3) \oplus \ldots) \oplus Bel_n.$$

This combination rule applies in two important strategic steps. The first is to commit the product of $m_1(H_i)$ and $m(S_j)$ to the possibility $H_i \cap S_j$, ie. $m(H_i \cap S_j) = m_1(H_i)m_2(S_j)$. The propositions which correspond to the subset H of Θ may have more than one probability mass committed exactly to H. By Eq. (5.3), we have the basic probability measure

$$m(H) = \sum_{\substack{i,j \\ H_i \cap S_j = H}} m_1(H_i)m_2(S_j).$$

One difficulty here is that some of the intersections of the focal elements H_i of Bel_1 and S_j of Bel_2 may be committed to an empty subset, ie. $H_i \cap S_j = \emptyset$. In such a case,

$$0 < \sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i)m_2(S_j) < 1.0.$$

So, the second step is to "discard" all probability measures committed to $\emptyset$ by normalizing the probability measures. This is done by multiplying those not committed to $\emptyset$ by a factor

$$\left(1 - \sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i)m_2(S_j)\right)^{-1}.$$

Dempster's rule of combination provides a method for changing prior opinions in the light of new evidence; one can construct belief functions to represent the new

evidence and combine them with the 'prior' belief functions, that is, with the belief functions that represent the prior knowledge. This method is clearly symmetrical (Shafer, 1976) and deals symmetrically with new and old evidence on which one's prior knowledge is based. Combined evidence is represented by new belief functions, and the result of combination does not depend on which evidence is old or on which evidence is new. Theoretically, any number of belief functions can be combined in any order.

One important requirement in using the Dempster's rule of combination to combine two or more belief functions is that the evidence represented by the belief functions must be independent. Shafer (1984) suggested that the problem of dependent evidence can be resolved by reframing Θ. We will see in the next chapter that dependent evidence can be dealt with by reframing in the knowledge-based system.

5.3 Representation of Exploration Data

Suppose we have an exploration area Ψ and an X-set

$$X = \{D_1, D_2, \ldots, D_n\},$$

where $D_i (i = 1, 2, \ldots, n)$ is a D-set in Ψ. We further assume that all D-sets are evidentially independent. Upon deciding an exploration target, a set of all possibilities Θ, a frame of discernment which recognizes the propositions related to the exploration target,

$$\Theta = \{H_1, H_2, \ldots, H_m\}$$

can be established. Moon (1990) called it a set of environmental possibilities. The propositions correspond to the subsets of Θ and the set of all subsets is denoted as 2^{Θ}. Belief functions can be constructed over Θ to represent D_i $(i = 1, 2, \ldots, n)$ towards

the exploration target. This operation, in terms of set-theoretic representation, is an $X - P$ mapping.

A complete set 2^{Θ_i} which corresponds to the propositions related to the target based on D_i, may be defined as

$$2^{\Theta_i} = \{\emptyset, T_i, \overline{T}_i, \Theta_i\}$$

where

$\emptyset$: an empty set,

T_i: the target exists based on D_i,

$\overline{T}_i$: the target does not exist based on D_i,

Θ_i: $\{T_i, \overline{T}_i\}$.

Probability numbers may then be estimated using an $X - P$ mapping G_i

$$\begin{aligned} m_i(T_i) &= G_i(D_i^1), \\ m_i(\overline{T}_i) &= G_i(D_i^2), \\ m_i(\Theta_i) &= 1 - m_i(T_i) - m_i(\overline{T}_i) \end{aligned}$$

and

$$m_i(\emptyset) = 0,$$

where D_i^1 and D_i^2 are subsets of D_i; $m_i(T_i)$ is the degree of belief (lower probability) that a target exists based on D_i. $m_i(\overline{T}_i)$ is the degree of disbelief or degree of belief that the target does not exist. $m_i(\Theta_i)$ is the degree of uncertainty based on D_i. The plausibility or upper probability is defined as

$$Pl(T_i) = m_i(T_i) + m_i(\Theta_i)$$

or

$$Pl(T_i) = 1 - m_i(\overline{T}_i).$$

There are two independent probabilities to be committed in this $X-P$ mapping. In some cases, one may only know $m_i(T_i)$ or $m_i(\overline{T}_i)$. The remainder can be simply committed to $m_i(\Theta_i)$. In the case that no belief or disbelief can be committed to $m_i(T_i)$ or $m_i(\overline{T}_i)$ based on D_i, one has $m_i(\Theta_i)=1$. This situation is the same as the case with no data. Similarly, n belief functions $Bel_1, Bel_2, \ldots Bel_n$ from $D_1, D_2, \ldots D_n$ can be obtained by $X-P$ mappings over the same frame of discernment for the same target.

The $X-P$ mappings can be performed by a statistical method if the exploration area is well surveyed and there are sufficient target occurrences in the area, or there exists a geologically compatible control area. The $X-P$ mapping by a statistical method is based on traditional probability theory and it is called traditional probabilistic $X-P$ mapping in this research. When the exploration area is less explored and there is no control area available, the $X-P$ mapping has to rely mainly on the explorationists' knowledge and expertise and can be called expert $X-P$ mapping.

5.3.1 Traditional Probabilistic $X-P$ Mapping

Suppose we have a D-set D_i from the X-set in a well developed exploration area Ψ, in which mineral deposits are discovered. In the D-set D_i, there are m attributes. For example, the attributes in a geological map may include felsic intrusions, mafic volcanics, etc.; a total magnetic map may include very high, medium anomaly, etc. The exploration area can be divided into N cells or squares of equal area. The number of squares having an attribute Q can be denoted as N_Q. N_{QT} denotes the number of squares with an attribute Q and a target mineral deposit. N_Q and N_{QT} can be obtained by counting the number of squares inside Ψ. Then $m_i(T_i)$ is given by

$$m_i(T_i)=\frac{N_{QT}}{N_Q}.$$

To determine $m_i(\overline{T}_i)$, one must know the areas in which one can conclude, based on sound evidence, that no target mineral deposit exists. If the number of squares in which no target mineral deposit exists can be counted and it can be denoted as $\overline{N}_{QT}$, the probability measure of $\overline{T}_i$ is

$$m_i(\overline{T}_i) = \frac{\overline{N}_{QT}}{N_Q}$$

and the uncertainty is defined as

$$m_i(\Theta_i) = 1 - \frac{N_{QT}}{N_Q} - \frac{\overline{N}_{QT}}{N_Q}.$$

In the case that $\overline{N}_{QT}$ is not available, which means that one does not have enough evidence or is uncertain about it, the belief function Bel_i can still be constructed by committing the remainder of $m_i(T_i)$ to $m_i(\Theta_i)$,

$$\begin{aligned} m_i(T_i) &= \frac{N_{QT}}{N_Q}, \\ m_i(\Theta_i) &= 1 - \frac{N_{QT}}{N_Q}, \\ m_i(\overline{T}_i) &= 0, \\ m_i(\emptyset) &= 0. \end{aligned}$$

These probability measures define a belief function Bel_i and can be combined with other belief functions. Unlike the representation using the Bayesian probability theory, one does not have to commit all remaining squares, $N_Q - N_{QT}$, in which one is trying to find mineral deposits, to $\overline{N}_{QT}$ which represents the squares in which there are no such mineral deposit. Therefore, the belief function theory provides a more natural and consistent basis of representing geological exploration information.

Another advantage of the belief function representation is that the total number of squares in Ψ is not used to determine the basic probability values for Bel_i. This means that a D-set with incomplete spatial coverage, which frequently happens in

exploration projects, can be integrated with others. If there exists a control area, the belief functions can be borrowed from it or refined by combining the data in the exploration area with the data in the control area. This combination is made possible by the fact that the total number of squares in the test area is not used for determining the belief functions. If no sufficient target mineral deposits have been discovered and no control area is available, an expert $X - P$ mapping has to be employed.

5.3.2 Expert $X - P$ Mapping

The real usefulness of the expert $X - P$ mapping is that it can be used in an under-explored area. The basic probability numbers are first assigned by an explorationist according to the theories of mineral deposits and normal interpretation and/or evaluation of the data in Ψ, instead of statistical analysis of mineral occurrences and attributes of the D-sets. The basic frame of discernment and propositions can be the same as in the traditional probabilistic $X - P$ mapping.

Suppose one has an X-set in Ψ in which no discovered mineral deposit or only a very limited number of mineral deposits are known. Reviewing the whole X-set and the limited but valuable information from the mineral occurrences in Ψ, the explorationist can obtain an approximate idea about which D-sets would have stronger support towards the target proposition and which D-sets would have stronger support towards its negation. And by analyzing each $D_i \subset X$ $(i = 1, 2, \ldots, n)$ the explorationist should be able to establish a qualitative measure about which attributes of D_i would contribute stronger support towards the target proposition or to its negation. Having learned these two qualitative relative measures, the basic probability numbers can be assigned to $m_i(T_i)$, $m_i(\overline{T}_i)$ and $m_i(\Theta_i)$. These assigned basic probability numbers can then be justified together until the explorationist becomes confident with the relative distributions.

For instance, if the target proposition involves a volcanic-associated massive sulphide deposit, a detailed geological map of good precision would have stronger support towards the target proposition or its negation than an airborne magnetic survey map. In the geological map, the attributes of volcanic rocks should give a higher level of support than those of intrusive rocks. The strong anomalies in an airborne EM map will similarly have stronger support than weak anomalies. If these relative relations are correct, the absolute values assigned to $m_i(T_i)$, $m_i(\overline{T}_i)$ and $m_i(\Theta_i)$ do not necessarily have to be accurate. Consequently, the results of combining belief functions $Bel_1, Bel_2, \ldots, Bel_n$ are relative and represent the relative probability measures from the whole X-set.

Although this representation approach relies on the explorationists' knowledge and expertise, it allows individual interpretations of D_i $(i = 1, 2, \ldots, n)$ to be pooled together to form a set of maps representing the target proposition from the whole X-set. These final maps outline, based on the available data and best to the explorationists' knowledge, favorable areas for further exploration.

5.4 Integration of Exploration Data Using Dempster's Rule of Combination

The belief functions estimated above can be integrated using the Dempster's rule of combination. Suppose we have two belief functions Bel_1 representing D_1 and Bel_2 representing D_2. The propositions for Bel_1 are H_1, H_2 and H_3, where H_1 is "the target exists from D_1", $H_2 = \overline{H}_1$ and $H_3 = \{H_1, H_2\}$. The corresponding basic probability measures are

$$\begin{aligned} m_1(H_1) &= b_1, \\ m_1(H_2) &= d_1, \end{aligned} \tag{5.15}$$

$$m_1(H_3) = u_1,$$

where b_1 corresponds to the degree of belief, d_1 corresponds to the degree of disbelief, and u_1 corresponds to the degree of uncertainty for the target proposition based on D_1. Similarly, let the propositions of Bel_2 be S_1, S_2 and S_3, where S_1 is "the target exists from D_2", $S_2 = \overline{S}_1$ and $S_3 = \{S_1, S_2\}$. The corresponding basic probability numbers are

$$\begin{aligned} m_2(S_1) &= b_2, \\ m_2(S_2) &= d_2, \\ m_2(S_3) &= u_2, \end{aligned} \tag{5.16}$$

where b_2 corresponds to the degree of belief, d_2 corresponds to the degree of disbelief, and u_2 the degree of uncertainty for the target proposition from D_2. All possible combinations and the probabilities committed to them by combining Bel_1 and Bel_2 are listed in Table 5.1. The intersection of two subsets H_1 and S_1, $H_1 \cap S_1$, in the top left corner of Table 5.1, for example, means that both D_1 and D_2 support the target proposition and $b_1 b_2$ is the probability committed to it. The focal elements of Bel which we are trying to establish for the target by combining D_1 and D_2 may be

$$\{T, \overline{T}, \Theta\}$$

where T is "the target exists from D_1 or D_2", $\overline{T}$ is its negation and $\Theta = \{T, \overline{T}\}$. Thus, referring to Table 5.1, the total probability committed to T is given by

$$\sum_{\substack{i,j \\ H_i \cap S_j = T}} m_1(H_i) m_2(S_j) = b_1 b_2 + u_1 b_2 + u_2 b_1,$$

where $b_1 b_2$ is the probability committed to $H_1 \cap S_1$ which means that both D_1 and D_2 support the proposition T, $u_1 b_2$ is the probability committed to $H_3 \cap S_1$ which means

	H_1	H_2	H_3
S_1	$H_1 \cap S_1$ $b_1 b_2$	$H_2 \cap S_1$ $d_1 b_2$	$H_3 \cap S_1$ $u_1 b_2$
S_2	$H_1 \cap S_2$ $b_1 d_2$	$H_2 \cap S_2$ $d_1 d_2$	$H_3 \cap S_2$ $u_1 d_2$
S_3	$H_1 \cap S_3$ $b_1 u_2$	$H_2 \cap S_3$ $d_1 u_2$	$H_3 \cap S_3$ $u_1 u_2$

Table 5.1: Propositions and their basic probability measures

that D_2 supports the proposition, and $u_2 b_1$ is the probability committed to $H_1 \cap S_3$ which means that D_1 supports the proposition. The total probability committed to $\overline{T}$ is given by

$$\sum_{\substack{i,j \\ H_i \cap S_j = \overline{T}}} m_1(H_i) m_2(S_j) = d_1 d_2 + u_1 d_2 + u_2 d_1,$$

where $d_1 d_2$ is the probability committed to $H_2 \cap S_2$ which means that both D_1 and D_2 support the negation of T, $u_1 d_2$ is the probability committed to $H_3 \cap S_2$ which means that D_2 supports the negation of T, and $u_2 d_1$ is the probability committed to $S_3 \cap H_2$ which means that D_1 supports the negation of T. The total probability committed to Θ is given by

$$\sum_{\substack{i,j \\ H_i \cap S_j = \Theta}} m_1(H_i) m_2(S_j) = u_1 u_2,$$

where $u_1 u_2$ is the probability committed to $H_3 \cap S_3$ which means the both D_1 and D_2 are uncertain about T. Because the intersections $H_1 \cap S_2$ and $H_2 \cap S_1$ do not exist, probabilities $b_1 d_2$ and $b_2 d_1$ are committed to an empty subset, namely,

$$\sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i) m_2(S_j) = b_1 d_2 + b_2 d_1.$$

According to the Dempster's rule of combination, the probability measures for Bel for the chosen proposition are given by

$$\begin{aligned} m(T) &= \frac{b_1 b_2 + b_2 u_1 + b_1 u_2}{1 - b_1 d_2 - b_2 d_1}, \\ m(\overline{T}) &= \frac{d_1 d_2 + d_1 u_2 + d_2 u_1}{1 - b_1 d_2 - b_2 d_1}, \end{aligned} \tag{5.17}$$

and

$$m(\Theta) = \frac{u_1 u_2}{1 - b_1 d_2 - b_2 d_1}.$$

Clearly, the above defined probability measures satisfy

$$m(T) + m(\overline{T}) + m(\Theta) = 1.$$

In the case that d_1 cannot be properly estimated or assigned for Bel_1 based on the given data and knowledge, one can assign $d_1 = 0$. The remainder of $m(H_1)$, $1 - m(H_1)$, is committed to $m(\Theta)$. Thus, the corresponding formulae for eg. (5.17) become

$$
\begin{aligned}
m(T) &= \frac{b_1 b_2 + b_1 u_2 + b_2 u_1}{1 - b_1 d_2}, \\
m(\overline{T}) &= \frac{d_2 u_1}{1 - b_1 d_2}, \qquad (5.18)
\end{aligned}
$$

and

$$
m(\Theta) = \frac{u_1 u_2}{1 - b_1 d_2}.
$$

In the case that both d_1 and d_2 cannot be properly estimated or assigned, one can assign $d_1 = 0$ and $d_2 = 0$. Thus the combination rule becomes

$$
\begin{aligned}
m(T) &= b_1 b_2 + b_1 u_2 + b_2 u_1, \\
m(\overline{T}) &= 0, \qquad (5.19)
\end{aligned}
$$

and

$$
m(\Theta) = u_1 u_2.
$$

The Dempster's rule of combination is symmetric. The belief functions which represent $D_3, \ldots D_n$ can then be combined one after another by repeatedly applying Eq. (5.17), (5.18), or (5.19).

The whole procedure of representation and integration of exploration data using the belief function theory can be, in terms of set-theoretic representation, expressed as

$$\left.\begin{array}{l} D_1 \xrightarrow{X-P} Bel_1 \\ D_2 \xrightarrow{X-P} Bel_2 \\ \vdots \\ D_n \xrightarrow{X-P} Bel_n \end{array}\right\} \xrightarrow{DRC} Bel, \tag{5.20}$$

where DRC means Dempster's rule of combination.

5.5 Test Examples

The above described approach is tested using the real exploration data from both Farley Lake (Figures 2.4-2.12) and Snow Lake area (Figures 2.19-2.17). There are only a few mapped occurrences of iron formation and base metal deposits in the Farley Lake area. In the southwest part of the Snow Lake test area, there are some base metal occurrences but they are very limited and most of the D-sets do not cover the locations of these mineral occurrences. The expert $X-P$ mapping has to be used in both areas.

5.5.1 Farley Lake Test Area

The target propositions for this test area are "there is a base metal deposit" and "there is an iron formation deposit". The propositions for the base metal from D_i $(i = 1, 2, \ldots, 9)$ are

T_{Bi} = "a base metal deposit exists from D_i",

$\overline{T}_{Bi}$ = "a base metal deposit does not exists from D_i",

$\Theta_{Bi} = \{T_{Bi}, \overline{T}_{Bi}\}$.

The basic probability numbers $m_i(T_{Bi})$, $m_i(\overline{T}_{Bi})$ and $m_i(\Theta_{Bi})$ assigned to the attributes of D_i are listed in Table 5.2. The propositions for the iron formation from

D_i $(i = 1, 2, \ldots, 9)$ are

$$T_{Ii} = \text{“an iron formation exists from } D_i\text{”},$$

$$\overline{T}_{Ii} = \text{“an iron formation does not exists from } D_i\text{”},$$

$$\Theta_{Ii} = \{T_{Ii}, \overline{T}_{Ii}\}.$$

The basic probability assignments $m_i(T_{Ii})$, $m_i(\overline{T}_{Ii})$ and $m_i(\Theta_{Ii})$ assigned to the attributes of D_i are listed in Table 5.3.

The belief functions are combined using Eq. (5.17) to estimate new belief functions which represent the total belief from pooling all the D-sets for the chosen exploration targets. The focal elements of the propositions being established are

$$T_I = \text{“an iron formation deposit exists from the X-set”},$$

$$\overline{T}_I = \text{“an iron formation does not exists from the X-set”},$$

$$\Theta_I = \{T_I, \overline{T}_I\},$$

for the iron formation and

$$T_B = \text{“a base metal deposit exists from the X-set”},$$

$$\overline{T}_B = \text{“a base metal deposit does not exists from the X-set”},$$

$$\Theta_B = \{T_B, \overline{T}_B\},$$

for the base metal.

The resulting distribution of belief functions is plotted in grey level plots. Figure 5.1 shows the spatial distribution of $m(T_I)$, the degree of belief or support to which T_I is true. The high support, $m(T_I)$, is found in the west central part of the test area, in which ore bodies with up to 95 percent pyrrhotite were discovered by drilling (Chevillard and Genaile, 1970). The low support is found in the lake areas and the areas in which we have less or no information. Figure 5.2 shows the spatial distribution of $m(\overline{T}_I)$, the degree of disbelief to which T_I cannot be believed according to the

D-set D_i	$m_i(T_{Bi})$	$m_i(\overline{T}_{Bi})$	$m_i(\Theta_{Bi})$
Ground EM Data			
no data	0.0	0.0	1.0
< 4	0.05	0.10	0.85
4 – 10	0.10	0.08	0.82
10 – 20	0.15	0.05	0.80
> 20	0.20	0.03	0.77
Airborne EM			
B	0.15	0.03	0.82
C	0.13	0.03	0.84
D	0.10	0.04	0.86
E	0.08	0.04	0.88
Band	0.05	0.05	0.90
No anomaly	0.05	0.07	0.88
No data	0.0	0.0	1.0
Ground Resistivity (ohm-m)			
< 100	0.25	0.03	0.72
100 – 500	0.17	0.05	0.78
500 – 1000	0.10	0.08	0.82
> 1000	0.05	0.10	0.85
no data	0.0	0.0	1.0
IP Chargeability Data (mV/V)			
> 40	0.25	0.03	0.72
20 – 40	0.20	0.05	0.75
6 – 20	0.10	0.08	0.82
< 6	0.05	0.10	0.85
no data	0.0	0.0	1.0
Aeromagnetic Data (γ)			
> 3000	0.05	0.08	0.87
500 - 3000	0.15	0.05	0.80
< 500	0.05	0.08	0.87

Table 5.2: Basic Probability Measure for a Base Metal Deposit (Farley Lake).

D-set D_i	$m_i(T_{Bi})$	$m_i(\overline{T}_{Bi})$	$m_i(\Theta_{Bi})$
VLF EM Data (Station: NNS)			
> 80	0.20	0.03	0.77
50 – 80	0.15	0.04	0.81
20 – 50	0.10	0.08	0.82
< 20	0.05	0.10	0.85
No data	0.0	0.0	1.0
VLF EM Data (Station: NLK)			
> 80	0.20	0.03	0.77
50 – 80	0.15	0.04	0.81
20 – 50	0.10	0.08	0.82
< 20	0.05	0.10	0.85
No data	0.0	0.0	1.0
Airborne INPUT EM Data			
No anomaly	0.05	0.07	0.88
Areal Anomaly	0.05	0.05	0.90
Band 2	0.09	0.05	0.86
Band 3 and 4	0.12	0.04	0.84
Band 5 and 6	0.15	0.03	0.82
Bedrock Geological Map			
Felsic Intrusive	0.05	0.10	0.85
Basalt – Andesite	0.15	0.05	0.80
Iron Rich Rocks	0.05	0.10	0.85
Picrite	0.05	0.10	0.85
Mafic Intrusives	0.15	0.05	0.80

Table 5.2: Basic Probability Measures for a Base Metal Deposit (Farley Lake). $m_i(T_{Bi})$–degree of belief for a base metal deposit from D_i; $m_i(\overline{T}_{Bi})$–degree of disbelief for a base metal deposit from D_i; $m_i(\Theta_{Bi})$–degree of uncertainty for a base metal from D_i.

D-set D_i	$m_i(T_{Ii})$	$m_i(\overline{T}_{Ii})$	$m_i(\Theta_{Ii})$
Ground EM Data			
No data	0.0	0.0	1.0
< 4	0.05	0.10	0.85
4 – 10	0.15	0.08	0.77
10 – 20	0.19	0.05	0.76
> 20	0.23	0.03	0.74
Airborne EM			
B	0.15	0.03	0.82
C	0.13	0.04	0.83
D	0.10	0.05	0.85
E	0.08	0.05	0.87
Band	0.05	0.05	0.90
No Anomaly	0.05	0.10	0.85
No data	0.0	0.0	1.0
Ground Resistivity (ohm-m)			
< 100	0.20	0.03	0.77
100 – 500	0.15	0.05	0.80
500 – 1000	0.10	0.05	0.85
> 1000	0.05	0.10	0.85
No data	0.0	0.0	1.0
IP Chargeability Data (mV/V)			
> 40	0.20	0.03	0.77
20 – 40	0.15	0.05	0.80
6 – 20	0.10	0.05	0.85
< 6	0.05	0.10	0.85
No data	0.0	0.0	1.0
Aeromagnetic Data (γ)			
> 3000	0.22	0.05	0.73
500 - 3000	0.15	0.10	0.75
< 500	0.05	0.15	0.80

Table 5.3: Basic Probability Measures for an Iron Formation (Farley Lake).

D-set D_i	$m_i(T_{Ii})$	$m_i(\overline{T}_{Ii})$	$m_i(\Theta_{Ii})$
VLF EM Data (Station: NNS)			
> 80	0.15	0.03	0.82
50 – 80	0.10	0.05	0.85
20 – 50	0.05	0.05	0.90
< 20	0.03	0.10	0.87
no data	0.0	0.0	1.0
VLF EM Data (Station: NLK)			
> 80	0.15	0.03	0.82
50 – 80	0.10	0.05	0.85
20 – 50	0.05	0.05	0.90
< 20	0.03	0.10	0.87
No data	0.0	0.0	1.0
Airborne INPUT EM Data			
No Anomaly	0.05	0.10	0.85
Areal Anomaly	0.05	0.05	0.90
Band 2	0.08	0.05	0.87
Band 3 and 4	0.10	0.05	0.85
Band 5 and 6	0.15	0.05	0.80
Bedrock Geological Map			
Felsic Intrusive	0.05	0.15	0.80
Basalt – Andesite	0.10	0.05	0.85
Iron Rich Rocks	0.25	0.05	0.70
Picrite	0.08	0.05	0.87
Mafic Intrusives	0.10	0.05	0.85

Table 5.3: Basic Probability Measures for an Iron Formation (Farley Lake). $m_i(T_{Ii})$–degree of belief for an iron formation from D_i; $m_i(\overline{T}_{Ii})$–degree of disbelief for an iron formation from D_i; $m_i(\Theta_{Ii})$–degree of uncertainty for an iron formation from D_i.

combined data. The high disbelief for an iron formation is found in the southeast part of the test area. The uncertainty $m(\Theta_I)$ plot (Figure 5.3) shows the degree to which the target proposition is uncertain. The higher uncertainty indicates that there is less or less efficient evidence in the X-set with respect to the given target and further data acquisition may be necessary. The plausibility is mathematically the sum of support and uncertainty. High plausibility is expected in the areas where there is either high support or high uncertainty or where both support and uncertainty are high. In practice, it can be interpreted as the upper boundary of probability which may be reached if mathematically sufficient evidence is acquired. Relatively high plausibility is found in the west central area where the support is high and in some of the areas where there is less data (Figure 5.4).

Similarly, for the target of base metal deposits, the spatial distribution plots of $m(T_B)$, $m(\overline{T}_B)$, $m(\Theta_B)$ and $m(T_B) + m(\Theta_B)$ correspond to degree of belief, disbelief, uncertainty and plausibility (Figures 5.5, 5.6, 5.7 and 5.8) towards the target proposition that there exists a base metal deposit from the X-set. The most favorable area for base metal deposits predicted by these plots is found in the west central area (Figure 5.5). The high disbelief is in the areas where there is more data coverage but some of them, such as chargeability, resistivity, etc. fail to show anomalies (Figure 5.6). A relatively high level of uncertainty is found in the areas where there is less or less efficient information (Figure 5.7). Relatively low uncertainty is found in the west central area where there is more data coverage. The high plausibility region can be found in areas with less data or with relatively high support.

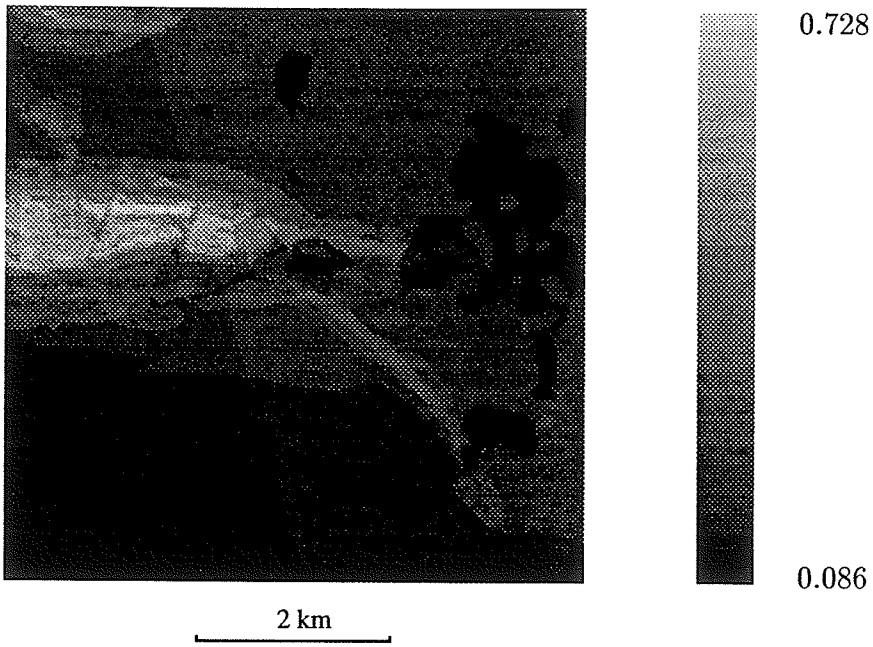

Figure 5.1: Support ($m(T_I)$) Map for an Iron Formation Deposit (Farley Lake).

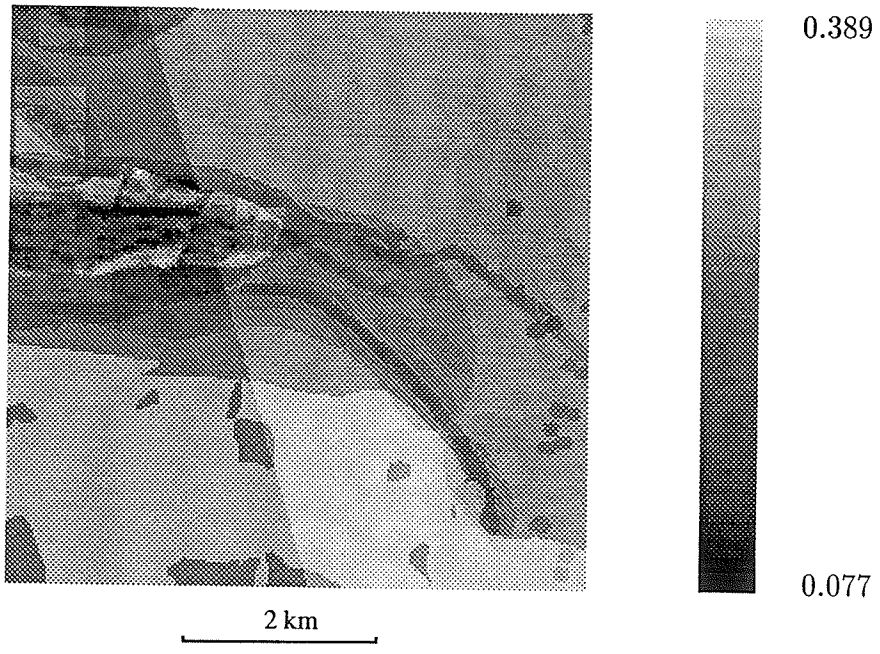

Figure 5.2: Disbelief ($m(\overline{T}_I)$) Map for an Iron Formation Deposit (Farley Lake).

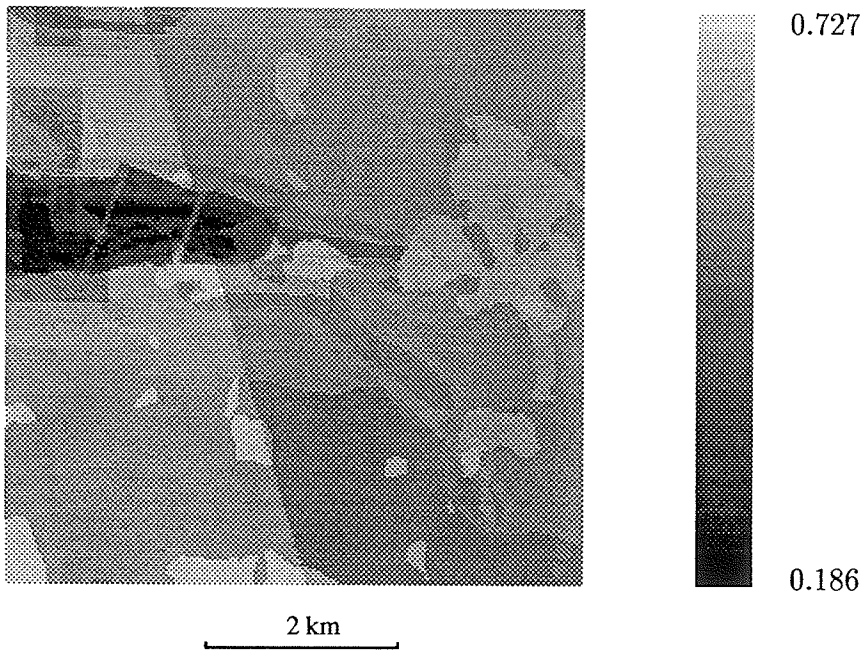

Figure 5.3: Uncertainty ($m(\Theta_I)$) Map for an Iron Formation Deposit (Farley Lake).

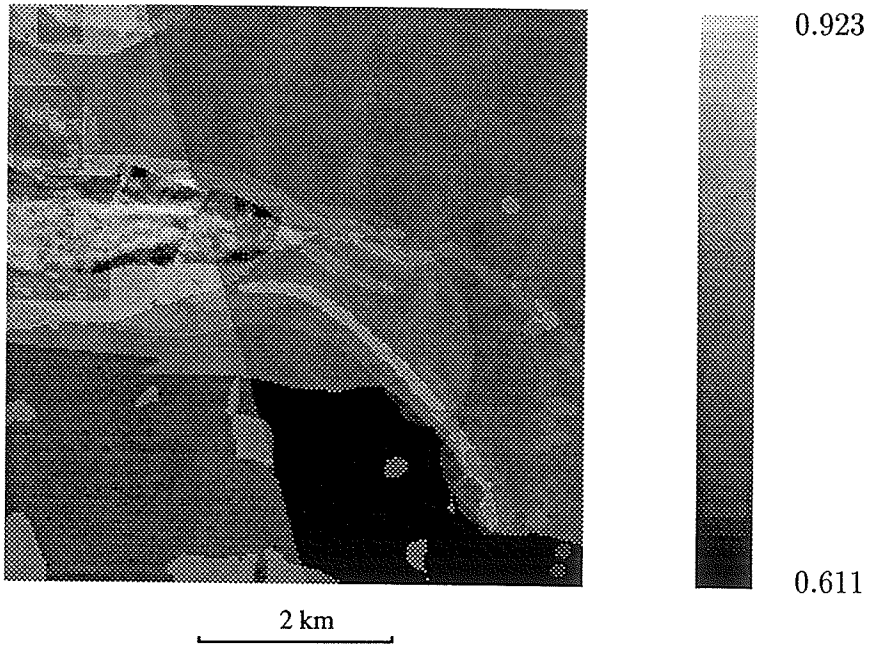

Figure 5.4: Plausibility ($m(T_I) + m(\Theta_I)$) Map for an Iron Formation Deposit (Farley Lake).

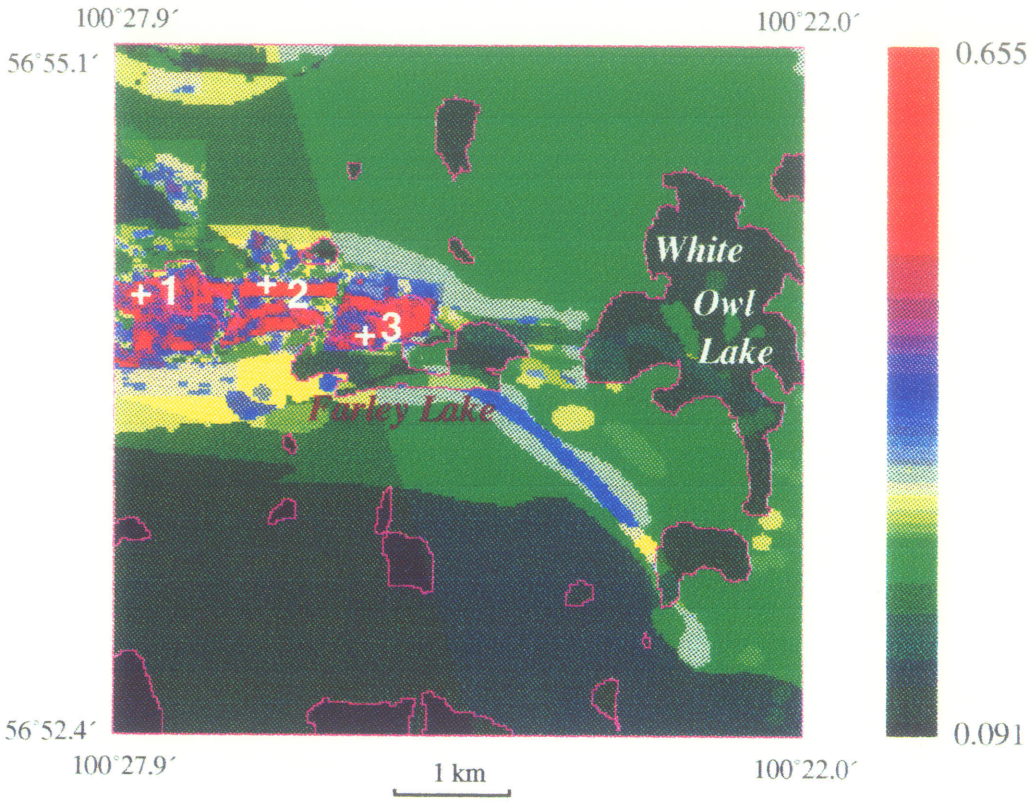

Figure 5.5: Support ($m(T_B)$) Map for a Base Metal Deposit (Farley Lake). 1 – Iron; 2, 3 – Gold.

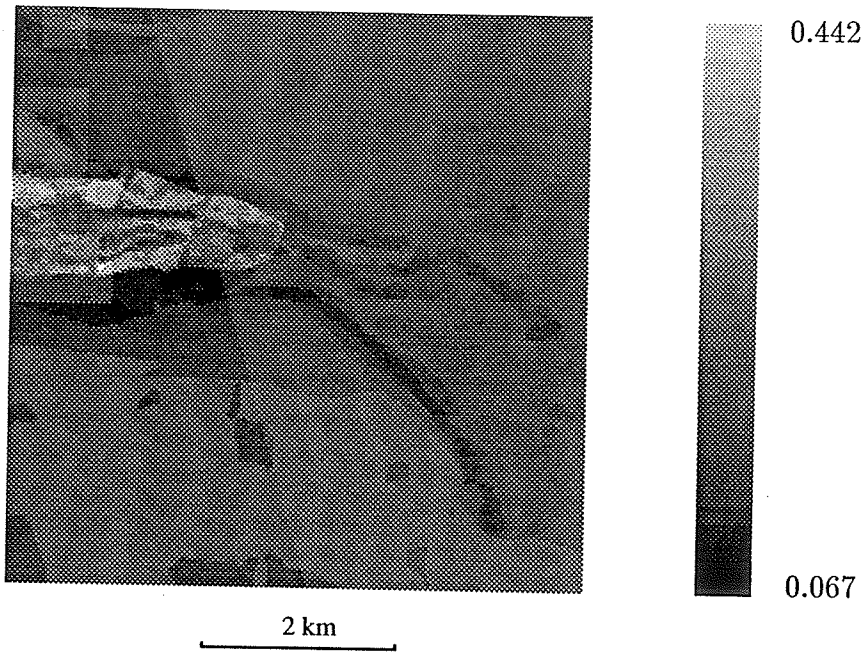

Figure 5.6: Disbelief ($m(\overline{T}_B)$) Map for a Base Metal Deposit (Farley Lake).

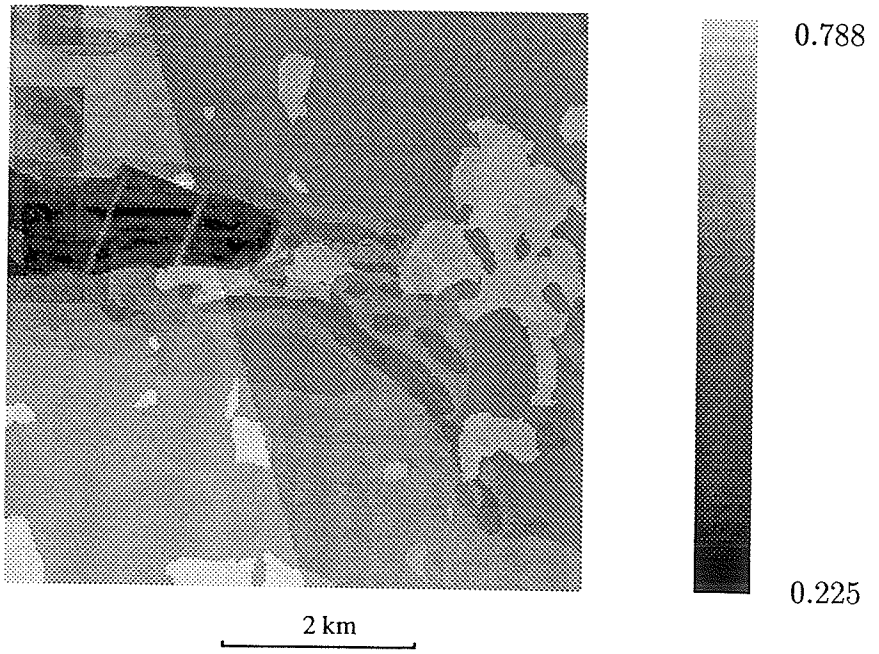

Figure 5.7: Uncertainty ($m(\Theta_B)$) Map for a Base Metal Deposit (Farley Lake).

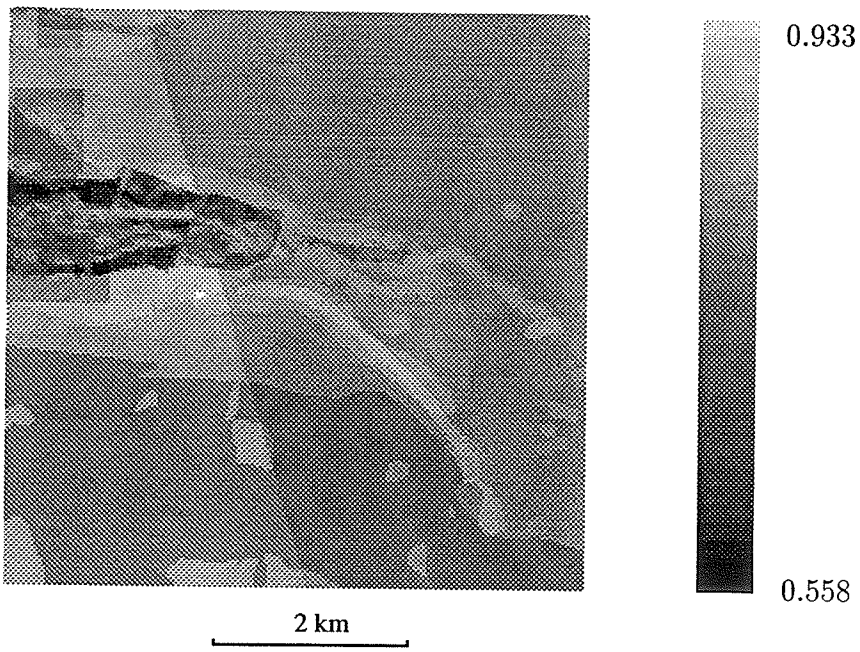

Figure 5.8: Plausibility $(m(T_B) + m(\Theta_B))$ Map for a Base Metal Deposit (Farley Lake).

5.5.2 Snow Lake Test Area

The target proposition tested in this area is "there is a base metal deposit". The corresponding propositions for different D-sets in this area are the same as in the Farley Lake test area. The basic probability numbers assigned to the attributes of the D-sets are listed in Table 5.4.

As in the Farley Lake area, the focal elements of the target proposition being established are

T_B = "a base metal deposit exists from the X-set",

$\overline{T}_B$ = "a base metal deposit does not exist from the X-set",

$\Theta_B = \{T_B, \overline{T}_B\}$.

D-set D_i	$m_i(T_{Bi})$	$m_i(\overline{T}_{Bi})$	$m(\Theta_{Bi})$
Airborne EM Data			
Very strong anomaly	0.15	0.01	0.84
Strong anomaly	0.12	0.03	0.85
Medium Anomaly	0.10	0.05	0.85
Week anomaly	0.07	0.07	0.86
No anomaly	0.02	0.08	0.90
No data	0.0	0.0	1.0
Ground EM Data			
Strong anomaly	0.25	0.01	0.74
Medium anomaly	0.20	0.03	0.77
Week anomaly	0.15	0.05	0.80
No anomaly	0.03	0.10	0.87
No data	0.0	0.0	1.0
Ground Magnetic Data			
Strong anomaly	0.15	0.03	0.82
Week anomaly	0.10	0.05	0.85
No anomaly	0.02	0.08	0.90
No data	0.0	0.0	1.0
Geological Map			
Felsic intrusions	0.01	0.10	0.89
Mafic intrusions	0.05	0.05	0.90
Quartzofeldspathic gneiss & migmatite	0.01	0.10	0.89
Sandstone and conglomerate	0.06	0.04	0.90
Greywacke, siltstone & mudstone	0.08	0.04	0.88
Felsic volcanic rocks	0.20	0.05	0.75
Mafic to intermediate volcanic rocks	0.20	0.03	0.77
No data (lakes)	0.0	0.0	1.0

Table 5.4: Basic Probability Measures for a Base Metal Deposit (Snow Lake).

D-set D_i	$m_i(T_{Bi})$	$m_i(\overline{T}_{Bi})$	$m(\Theta_{Bi})$
Airborne Magnetic Data (γ)			
> 3000	0.05	0.10	0.85
2550 – 3000	0.10	0.02	0.88
< 2550	0.05	0.08	0.87

Table 5.4: Basic Probability Measures for a Base Metal Deposit (Snow Lake). $m_i(T_{Bi})$–degree of belief for a base metal deposit from D_i; $m_i(\overline{T}_{Bi})$–degree of disbelief for a base metal deposit from D_i; $m(\Theta_{Bi})$–degree of uncertainty from D_i.

The spatial distribution of the probabilities which correspond to degrees of support ($m(T_B)$), disbelief ($m(\overline{T}_B)$), uncertainty ($m(\Theta_B)$) and plausibility ($m(T_B) + m(\Theta_B)$) are plotted in Figures 5.9, 5.10, 5.11 and 5.12. From Figure 5.9, high support is closely related to the ground EM anomalies. This is because the ground EM anomaly map provides more important evidence than other D-sets for the base metal deposits in this area. The relatively high disbelief is found in the central test area (Figure 5.10). The uncertainty is relatively high in most of the test area (Figure 5.11). The areas in which EM anomalies are found have relatively low uncertainty. The low plausibility is in the central area corresponding to the area with relatively high disbelief (Figure 5.12). Note that these results are based on the X-set. Because most of the data do not spatially cover the southwestern portion of the test area where base metal deposits are found, the support in this area is not high. But disbelief is very low and uncertainty is very high in this area, which suggests that it is a promising area and more information is needed for making a more conclusive decision.

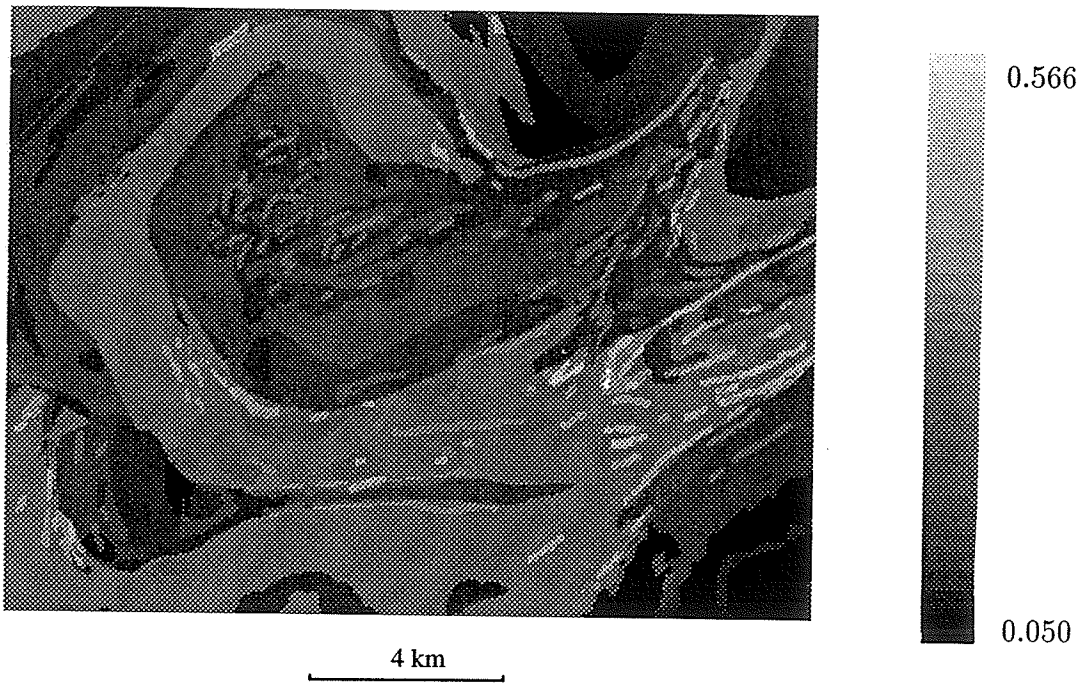

Figure 5.9: Support ($m(T_B)$) Map for a Base Metal Deposit (Snow Lake).

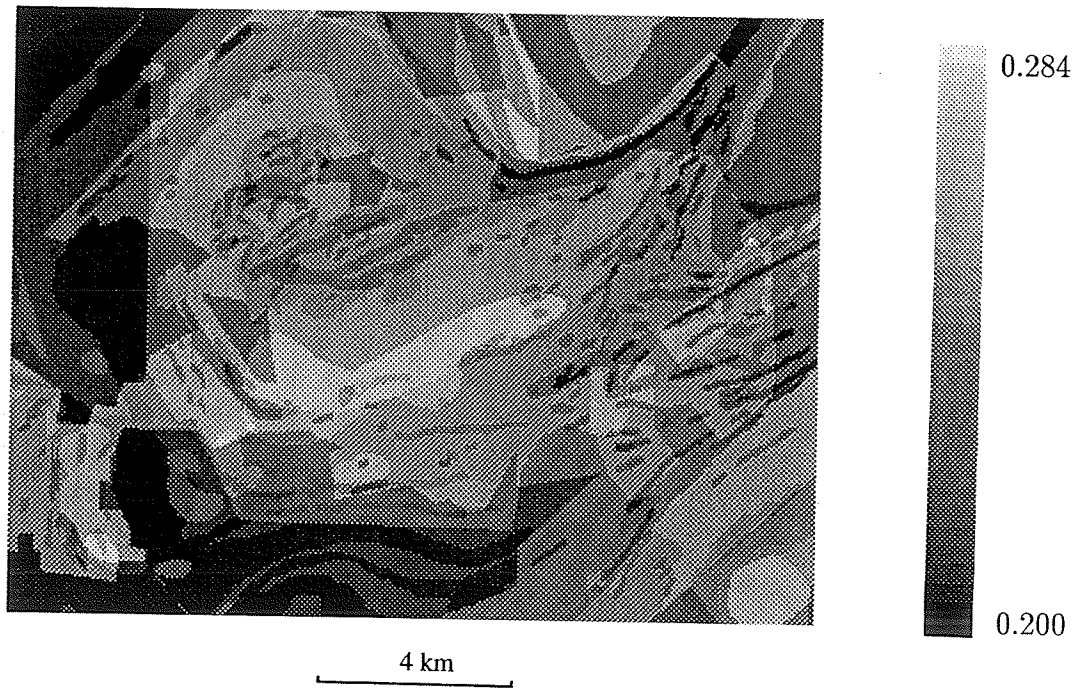

Figure 5.10: Disbelief ($m(\overline{T}_B)$) Map for a Base Metal Deposit (Snow Lake).

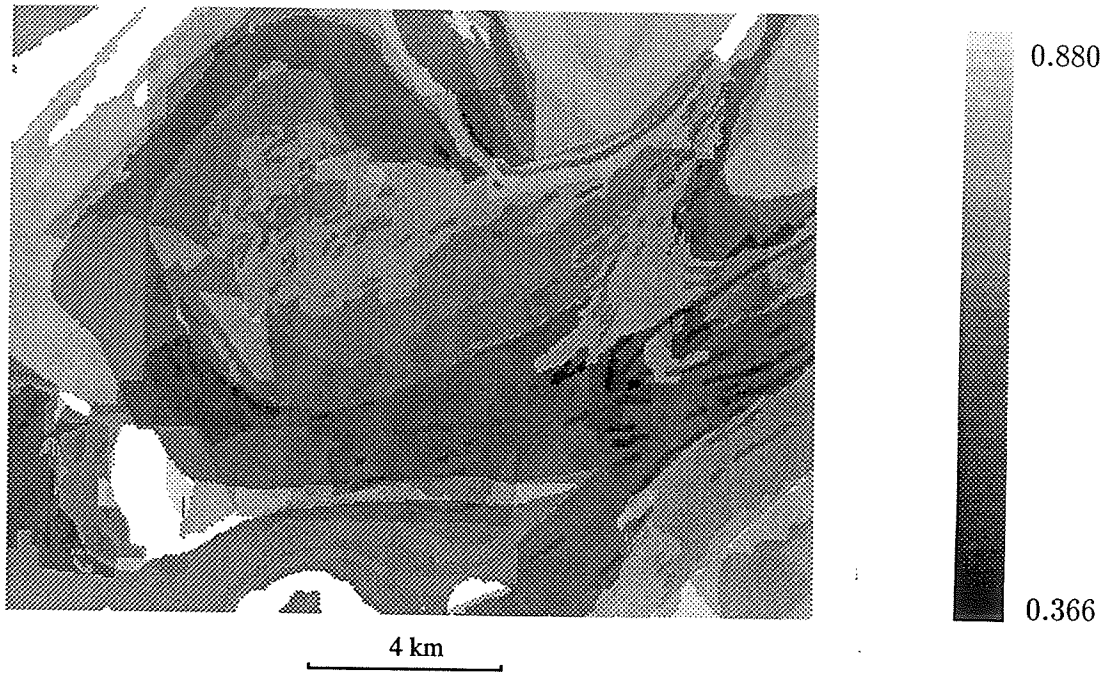

Figure 5.11: Uncertainty ($m(\Theta_B)$) Map for a Base Metal Deposit (Snow Lake).

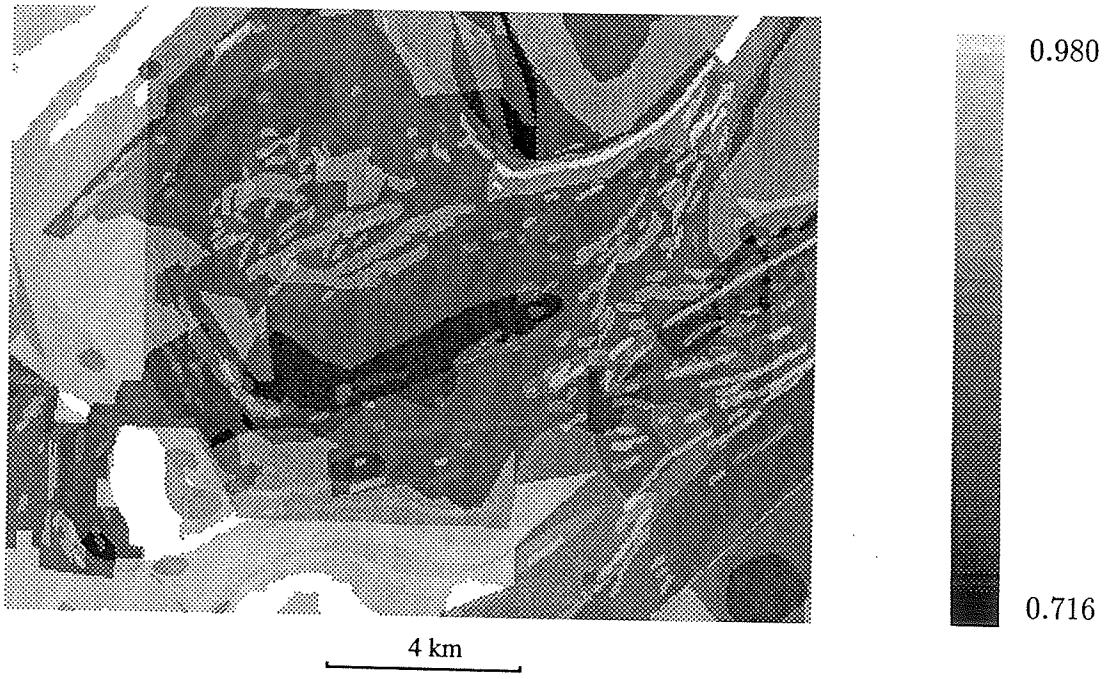

Figure 5.12: Plausibility ($m(T_B) + m(\Theta_B)$) Map for a Base Metal Deposit (Snow Lake).

5.6 Error Estimation and Degree of Conflict

All $X-P$ mappings contain experimental errors and uncertainties. They can propagate through the DRC towards the probabilities for the final target propositions. The Dempster-Shafer approach also provides quantities which can represent the conflict in the D-sets. By analyzing and modifying these quantities, a value which is called the degree of conflict can be specified. As defined, the degree of conflict has close resemblance to variance in the relative error approximation.

5.6.1 Error Propagation

The probability measures of Bel_1 and Bel_2 contain experimental and numerical errors. They can propagate through each step of the Dempster's rule of combination towards the final probability obtained by combining Bel_1 and Bel_2. Recall **5.2** where $m_1(H_i)$, $m_2(S_j)$, and $m(H)$ are defined, and let ϵ_{Hi} be the error term in $m_1(H_i)$ and ϵ_{Sj} in $m_2(S_j)$ respectively. By denoting

$$\begin{aligned} m_1(H_h) &= 1-\sum_{i=1}^{h-1} m_1(H_i) \\ m_2(S_l) &= 1-\sum_{j=1}^{l-1} m_2(S_j) \end{aligned}$$

and ignoring the second and higher terms of the Taylor expansion of Eq. (5.14), the following equation gives the error estimation ϵ_H associated with $m(H)$

$$\epsilon_H = \sum_{i=1}^{h-1} \frac{\partial m(H)}{\partial m_1(H_i)} \epsilon_{Hi} + \sum_{j=1}^{l-1} \frac{\partial m(H)}{\partial m_2(S_j)} \epsilon_{Sj}$$

$$
\begin{aligned}
= \frac{1}{1-k}\Bigg[& \sum_{i=1}^{h-1}\left(\sum_{\substack{j \\ H_i\cap S_j=H}} m_2(S_j) - \sum_{\substack{j \\ H_h\cap S_j=H}} m_2(S_j)\right)\epsilon_{Hi} \\
& + \sum_{j=1}^{l-1}\left(\sum_{\substack{i \\ H_i\cap S_j=H}} m_1(H_i) - \sum_{\substack{i \\ H_i\cap S_l=H}} m_1(H_i)\right)\epsilon_{Sj} \qquad (5.21) \\
& + m(H)\sum_{i=1}^{h-1}\left(\sum_{\substack{j \\ H_i\cap S_j=\emptyset}} m_2(S_j) - \sum_{\substack{j \\ H_h\cap S_j=\emptyset}} m_2(S_j)\right)\epsilon_{Hi} \\
& + m(H)\sum_{j=1}^{l-1}\left(\sum_{\substack{i \\ H_i\cap S_j=\emptyset}} m_1(H_i) - \sum_{\substack{i \\ H_i\cap S_l=\emptyset}} m_1(H_i)\right)\epsilon_{Sj}\Bigg]
\end{aligned}
$$

where k has been defined in Eq. (5.13).

If we take the errors from the $X - P$ mapping into consideration, the equation (5.15) which defines the belief function Bel_1 becomes

$$
\begin{aligned}
m_1(H_1) &= b_1 + \epsilon_{b1}, \\
m_1(H_2) &= d_1 + \epsilon_{d1}, \\
m_1(H_3) &= u_1 + \epsilon_{u1},
\end{aligned}
$$

where ϵ_{b1}, ϵ_{d1} and ϵ_{u1} are the error terms associated with $m_1(H_1)$, $m_1(H_2)$, and $m_1(H_3)$, and similarly, equation (5.16) which defines the belief function bel_2 becomes

$$
\begin{aligned}
m_2(S_1) &= b_2 + \epsilon_{b2}, \\
m_2(S_2) &= d_2 + \epsilon_{d2}, \\
m_2(S_3) &= u_2 + \epsilon_{u2},
\end{aligned}
$$

where ϵ_{b2}, ϵ_{d2} and ϵ_{u2} are the error terms corresponding to $m_2(S_1)$, $m_2(S_2)$ and $m_2(S_3)$. The error estimates for the target proposition corresponding to Eq. (5.17) are given

by

$$\epsilon_T = \frac{1}{1-k}\left[\left(m_2(S_3)+m(T)m_2(S_2)\right)\epsilon_{b1}+\left(m_1(H_3)+m(T)m_1(H_2)\right)\epsilon_{b2}\right.$$

$$\left.+\left(m(T)m_2(S_1)-m_2(S_1)\right)\epsilon_{d1}+\left(m(T)m_1(H_1)-m_1(H_1)\right)\epsilon_{d2}\right],$$

$$\epsilon_{\overline{T}} = \frac{1}{1-k}\left[\left(m_2(S_3)+m(\overline{T})m_2(S_2)\right)\epsilon_{d1}+\left(m_1(H_3)+m(\overline{T})m_1(H_2)\right)\epsilon_{d2}\right.$$

$$\left.\left(m(\overline{T})m_2(S_2)-m_2(S_2)\right)\epsilon_{b1}+\left(m(\overline{T})m_1(H_2)-m_1(H_2)\right)\epsilon_{b2}\right],$$

$$\epsilon_{\Theta_T} = \frac{1}{1-k}\left[\left(m_2(S_2)m(\Theta)-m_2(S_3)\right)\epsilon_{b1}+\left(m_1(H_2)m(\Theta)-m_1(H_3)\right)\epsilon_{b2}\right.$$

$$\left.\left(m(\Theta)m_2(S_1)-m_2(S_3)\right)\epsilon_{d1}+\left(m(\Theta)m_1(H_1)-m_1(H_3)\right)\epsilon_{d2}\right].$$

5.6.2 Degree of Conflict

While two belief functions are combined using Dempster's rule of combination, some of the probabilities may be committed to disjoint or contradictory subsets ($H_i \cap S_j = \emptyset$). The total probability committed to contradictory subsets is given by

$$k = \sum_{\substack{i,j \\ H_i \cap S_i = \emptyset}} m_1(H_i)m_2(S_j)$$

and this is the same as Eq. (5.13). The renormalization constant

$$K = \frac{1}{1-k}$$

increases with k and presents the extent of the conflict between Bel_1 and Bel_2. So, Shafer (1976) defined the weight of conflict as

$$\log K = -\log 1 - k.$$

In error evaluation, $\log K$ does represent the relative amount of conflict contained in the two bodies of evidence, but it is difficult to interpret physically because it does not have physically obvious relationships towards the probabilities committed to the non-empty subsets. The quantity k is used instead. If the two belief functions do not conflict, then $k = 0$; $k = 0.5$ means that the total probability committed to contradictory subsets is equal to the amount committed to the all other subsets. So, $k < 0.5$ means that

$$\sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i)m_2(S_j) < \sum_{H \subset \Theta} \sum_{\substack{i,j \\ H_i \cap S_j = H}} m_1(H_i)m_2(S_j) \tag{5.22}$$

and $k > 0.5$ means that

$$\sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i)m_2(S_j) > \sum_{H \subset \Theta} \sum_{\substack{i,j \\ H_i \cap S_j = H}} m_1(H_i)m_2(S_j). \tag{5.23}$$

The special case, $k = 1$, means that Bel_1 and Bel_2 flatly conflicts and $Bel_1 \oplus Bel_2$ does not exist.

The calculation of k can be extended to a collection of more than two belief functions

$$Bel_1, Bel_2, \ldots, Bel_{n+1}.$$

The total probability committed to contradictory subsets k is given by

$$k = 1 - \frac{1}{K_1 K_2 \ldots K_n}$$

where

$$K_i = \frac{1}{1 - k_i}$$

with

$$(i = 1, 2, \ldots, n)$$

is the renormalization constant for

$$(\ldots(Bel_1 \oplus Bel_2) \oplus \ldots Bel_{i-1}) \oplus Bel_i.$$

There are still two problems in the actual application of k in integration of exploration information. The first originates from the problem of incomplete spatial data coverage over the exploration areas. To resolve this problem, q is defined as

$$q = \frac{k}{n}. \tag{5.24}$$

Thus q is expected to balance the difference caused by varying spatial data coverage and may be interpreted as an approximation to the average of probability committed to empty sets.

Another problem is the information efficiency of data attributes in the same given data layer. For less efficient data attributes, uncertainty is high and in general $m(\Theta)$ is larger. The q consequently becomes smaller. A new quantity, which is called *the degree of conflict*, is defined as

$$\Gamma = \frac{q}{1 - m(\Theta)}. \tag{5.25}$$

The above formula that defines the degree of conflict reduces the effect of efficiency difference in a given D-set. So, Γ is an approximately balanced quantity and provides a spatial estimate of the degree of conflict for the combined data. Intuitively, this value, Γ, is related to the errors in the data. Erroneous data tend to induce the higher degree of conflict and the less erroneous data the lower degree of conflict. Its

functional role is like the variance in statistical analyses. The high degree of conflict tends to occur where some data layers show strong support towards the proposition but some of the data layers show strong support to its negation. Equations (5.24) and (5.25) are empirical relations and are expected, to a certain extent, to balance the effect of the number difference of the spatial data coverage and data efficiency difference. They tend to improve the spatial comparison of the conflict. The degree of conflict can be calculated both in the traditional probabilistic $X - P$ mapping and in the expert $X - P$ mapping.

The degree of conflict is calculated for both test areas. Figures 5.13 and 5.14 show the spatial distribution of degree of conflict for the proposition of a base metal and that of an iron formation in the Farley Lake area. In the west central part of the test area, both data accuracy and spatial data coverages are better and the uncertainty distribution is low. The degree of conflict is relatively low in the area for the proposition of an iron formation (Figure 5.14) but relatively high for the proposition of a base metal deposit (Figure 5.13). This shows that there is a higher potential for an iron formation deposit than for a base metal deposit although the belief distributions for both propositions are high in the area (Figures 5.1 and 5.5).

Figure 5.15 shows a spatial distribution of the degree of conflict for the proposition of a base metal deposit in the Snow Lake area. The black level in this plot means that Γ does not exist because there is only one data layer there. As mentioned in the last section, the locations of high belief are closely related to the ground EM anomalies (Figure 5.9). Some of the locations with high belief (Figure 5.9) show a low degree of conflict (Figure 5.15), which indicates that the combined data at these locations more consistently support the target proposition than those with a high degree of conflict.

The above described test further shows that the degree of conflict is a very important parameter for evaluating the final results. It also can be an important criterion

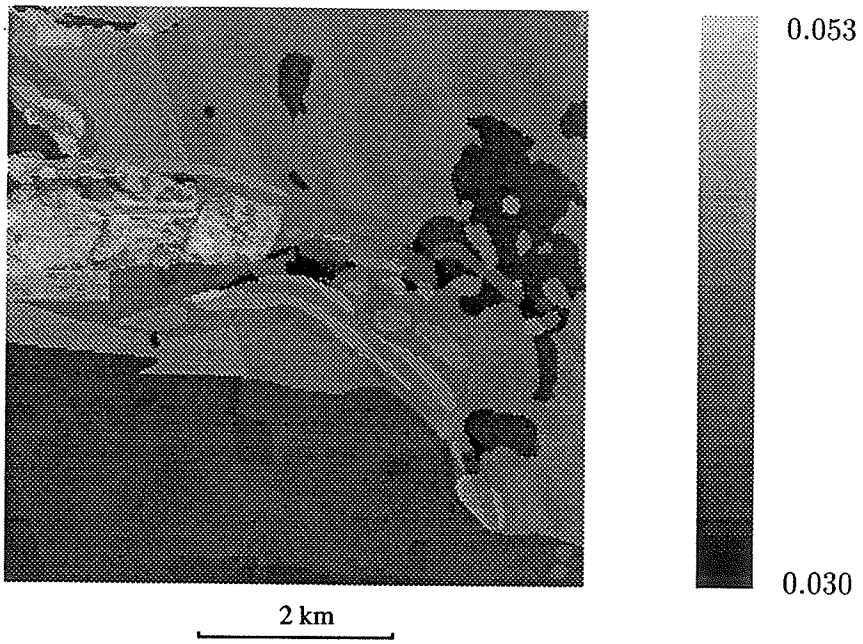

Figure 5.13: Degree of Conflict (Γ) Map for Base Metal (Farley Lake).

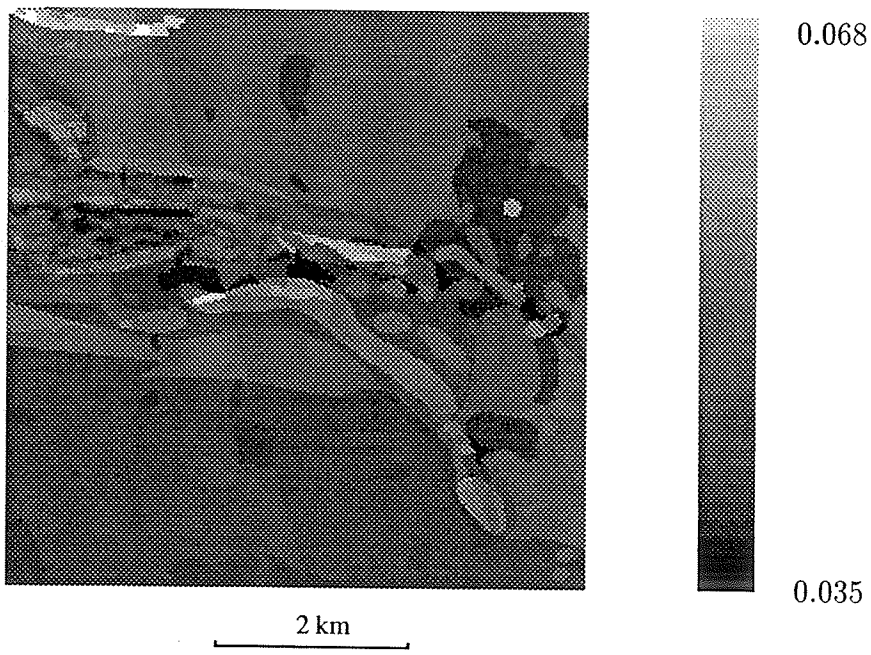

Figure 5.14: Degree of Conflict (Γ) Map for Iron Formation (Farley Lake).

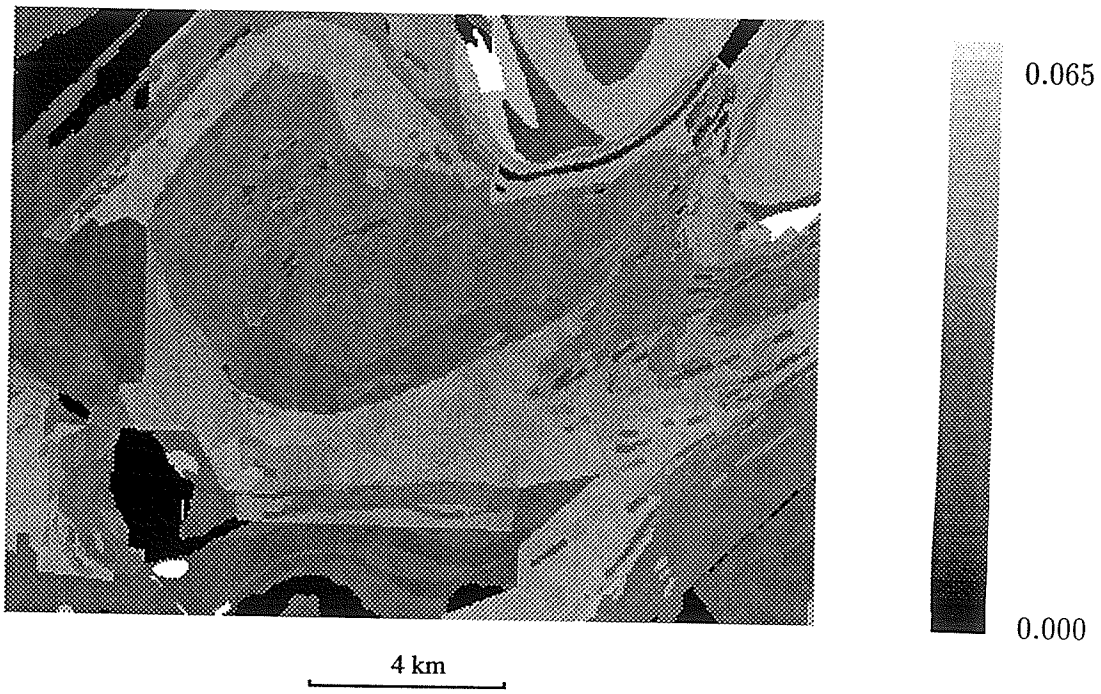

Figure 5.15: Degree of Conflict (Γ) Map for Base Metal (Snow Lake).

for accepting or denying a proposition in knowledge-based decision making processes.

5.7 Discussion and Conclusion

In non-renewable resource exploration, explorationists can never simply say "yes" or "no" to questions asking for specific exploration targets before it is observed directly or drilled. Uncertainty always exists when one tries to make a decision. The Dempster-Shafer (D-S) evidential belief function provides an adequate way to represent the intrinsically imprecise information and uncertain propositions. In the application of the D-S evidential belief function theory to represent exploration data, one does not have to commit the remainder of a probability measure to its negation. It allows one to represent the degree of uncertainty associated with data and propositions. This

capability is the major difference between this approach and the Bayesian approach. The D-S evidential belief function theory provides a more natural problem solving basis for realistic information representation than the traditional probability theories.

Another advantage of the D-S approach is its ability to handle the D-sets with incomplete coverage. In most non-renewable resource exploration projects, the D-sets are spatially incomplete and statistically unbalanced although the rate of data acquisition is alarmingly rapid. The D-S approach can handle this situation naturally and allows incomplete D-sets to be integrated with others, both in highly explored and underexplored areas.

The test results in the two test areas outline the most favorable target exploration areas. The known mineral deposit occurrences located in the high support areas predicted by this approach have independently confirmed the effectiveness of the methods utilizing the D-S evidential belief function. An exception is in the southwest part of the Snow Lake test area where there is insufficient data coverage. Consequently, the result shows low disbelief and high uncertainty, which suggests that further data acquisition should be carried out in this region. The outcome of this approach can provide more information for a proposition which allows one to carry out detailed analysis of low support, and to distinguish lack of support from support to its negation. Apparently, the D-S evidential belief function approach provides a more realistic picture of the target proposition in the exploration area.

One problem with the D-S approach is its stringent requirement for the evidential independence of the D-sets. Shafer (1984) suggested the dependent evidence can be handled by reframing the frame of discernment. This can be implemented in a knowledge-base system. Furthermore, the expert $X - P$ mapping also lends itself to an expert system approach. The D-S approach can thus be a very powerful tool for integrated resource exploration tasks.

Chapter 6

An Object-oriented and Map-based Expert System

An object-oriented and map-based prototype expert system has been developed in this research using programming language C^{++} for integrating different geoscience data sets for base metal exploration. The prototype system has been tested using real exploration D-sets from Farley Lake and Snow Lake test areas, Manitoba, Canada. The development is based on the set-theoretic representation and integration of spatial data, evidence representation, combination and uncertainty quantification described in the preceding chapters. The Dempster-Shafer evidential belief function theory (Dempster, 1967; Shafer, 1976) is utilized to manage uncertainties in both knowledge representation and uncertainty propagation. The object-oriented knowledge representation structure and the reasoning procedures for uncertainty propagation performed well for the chosen test examples. In addition to the expected advantages of the knowledge-based approach, physically known dependent evidence can be better dealt with in a knowledge-based approach.

The present system has 23 nodes (objects) and includes more than 150 rules. It consists of a knowledge base, inference engine and an object base (Figure 6.1). A set

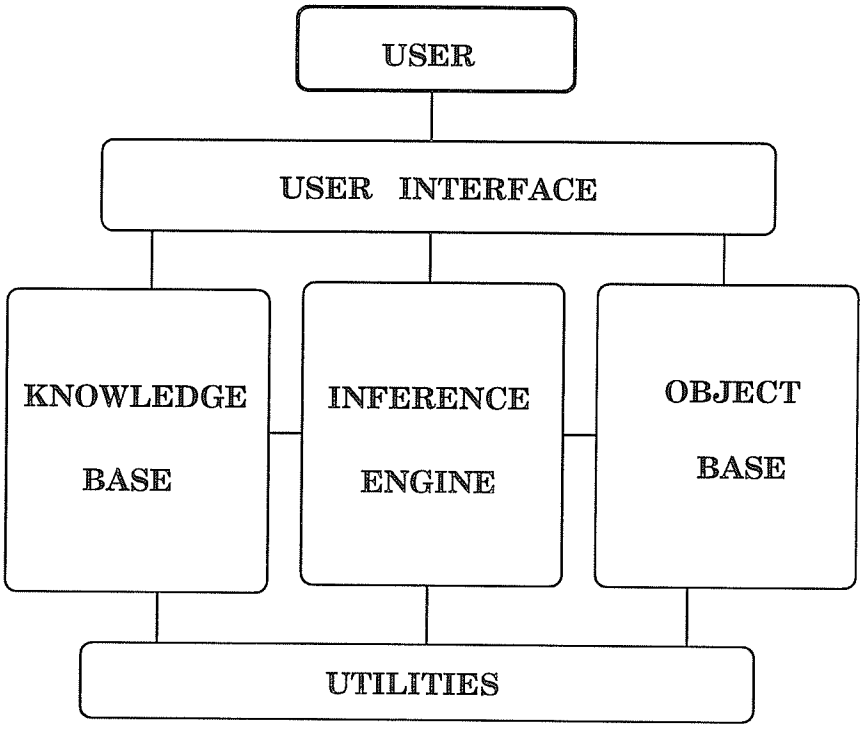

Figure 6.1: Block Diagram of the Prototype System

of utilities was developed to define and provide a necessary programming environment because the C^{++} is a relatively low-level programming language. The development is focused on knowledge representation structure, its corresponding inference mechanisms and object simulation of the data and related concepts, so that the user interface is simple and can also be used to acquire judgements about data from users.

In the following sections, several major problems in the application of expert system techniques to resource exploration will first be discussed. This will be followed by descriptions of object-oriented knowledge representation structure and its corresponding inference mechanisms. Finally the prototype expert system developed will be presented.

6.1 Some Problems in the Application of AI Techniques to Resource Exploration

The work which was carried out by Duda et al. at SRI in the late 1970's has been recognized as a pioneering work in the field of expert system research and development. It is also the first expert system for geoscience application. Numerous new systems have been developed and reported since then for resource exploration application. These systems were developed to varying degrees of completeness, from a mere concept to a completely functional research type, but very few have reached the production stage. Exploration applications are minor when they are compared to expert system applications in engineering and manufacturing, where thousands of systems are reported to be in production use today (Fox, 1990). Why has the expert system technique not been widely used in resource exploration? Aminzadeh (1991) summarized two reasons for the lack of success in oil exploration: (1) oil and gas exploration is in general highly multidisciplinary and (2) rules governing the exploration process are, for the most part, relatively more subjective. In this section, I will

briefly review some of the common problems in the resource exploration procedure itself and recent research and practices in AI applications targeted towards resource exploration.

Expert systems usually attain high levels of performance either in a narrow problem area (Waterman, 1986; Winston, 1984) or in a narrow and well-defined domain. But most tasks in non-renewable resource exploration are neither narrow nor well defined. Non-renewable resource exploration activities can in general be divided into two categories as described in Chapter 3. The first is tasks of acquiring and improving the X-set. Classical set theory is essentially the backbone of this category. A task in this category can in turn be divided into many subtasks such as airborne EM data acquisition, surficial geological mapping, seismic data processing, etc. There is often no interaction between these subtasks. The second category includes interpretation of the X-set to identify and isolate selected exploration targets. Although the D-sets can be interpreted individually, the final decision has to be based on the whole X-set. The tasks in this category are highly multidisciplinary and yet less adequately defined than those in the first. This fact may explain why most of the expert system applications to resource exploration are limited to specific domains, as presented by numerous authors such as Dulac (1991), Coppens (1991), Zheng and Simaan (1991) and Fang et al. (1991).

The tasks in the second category cannot be represented adequately in the framework of classical set theory. Most of the geoscience data are intrinsically imprecise although they are established in a framework of the classical set theory. In the interpretation of a D-set or D-sets, one can rarely answer "yes" or "no" to a proposition or exploration target, even when all the available data sets are judged to be of good quality. One can at best say vaguely "to some degree yes" or "to some degree no". Usually no clear cut solution or answer is available. This situation reflects the fact that information we gather cannot provide us with definite or absolute evidence.

Uncertainty management is thus an essential issue for expert system applications particularly in non-renewable resource exploration. Most expert systems developed and/or reported after PROSPECTOR deal little with uncertainty estimation. The multidisciplinary nature of non-renewable resource exploration tasks requires integrated approaches. Representation of imprecise information and non-absolute knowledge, uncertainty processing and inference mechanisms have unique roles in expert system applications to integrated exploration. Clearly, expert system applications in this category are considerably more difficult than many recently reported expert systems. This is one of the reasons why most AI systems in production use are in engineering and manufacturing rather than in non-renewable resource exploration.

Subjective reasoning is an important part of human knowledge processing and plays an essential role in many human reasoning processes. Such subjective reasoning can be based on information acquired from textbooks, scientific publications, and personal experiences from various other exploration areas. Of course, there is no problem to apply this type of knowledge to an exploration task. The important point however is to use this type of knowledge or reasoning process in a specific exploration project and to combine them with newly discovered knowledge which may be obtained directly from the project area and/or a geologically similar area.

Another problem is associated with correct interpretation of certainty factors estimated by an expert system. The results are less useful if they are difficult to interpret. If the output of a map-based system is maps, it will show different degrees of spatial favorabilities towards a specific exploration target. These maps show spatial distributions of specific information for the prospecting area. Even when certain pixel values appear to be meaningless, the overall distribution usually provides a relative favorability distribution towards the chosen exploration target.

6.2 Integration and Object-Oriented Programming

Object-oriented programming is a style of programming that is based on direct represention of physical objects and intellectual concepts. It is recognized by many AI researchers as the most appropriate and suitable vehicle for intelligent computation and is increasingly applied in real world problems.

Very recently, there appeared several reports on object-oriented programming being applied to solve geoscience problems. Among them are Miller (1990) who reported an object-oriented expert system under development at the United States Geological Survey, Dulac (1991) who built an object-oriented intelligent front-end for a seismic processing package and Lafue et al. (1991) who argued that an object-oriented knowledge representation system would be better for representing data and knowledge for reservoir analysis.

Given an exploration strategy, D-sets are usually acquired independently and usually interpreted separately for target parameters. One can even interpret one D-set without knowing how to interpret other D-sets. For example, an airborne magnetic D-set can be interpreted while he/she does not have to know how to interpret a gravity D-set. The rules that govern processing and interpretation of one specific D-set usually are not applicable for interpretation of a D-set from another survey. So, the first step of an integrated exploration project includes separate interpretation of D-sets for specific parameters relevant to specific exploration targets. The results of each interpretation can then be stored and represented in a specific format in which it can be combined with the interpretation from other D-sets. This process suggests that different program blocks can be built to process and interpret different D-sets, to store and represent their interpretation results, and to combine the interpretations.

Equally important are the propositions (concepts) which are necessary in a later

integrated analysis. The propositions receive evidence from D-sets or other propositions. The evidence is then processed and the results can be updated as new evidence for other propositions. These tasks require similar program blocks which can communicate with each other, can process evidence, and store the results.

An important functional requirement of the program blocks is the problem-solving capability which performs processing and communication steps intelligently. This further implies that the program blocks should be able to store intelligent knowledge. In summary, an intelligent integrated exploration system requires basic intelligent program blocks which have the capability to store information and knowledge, to process and interpret information, to create new information, and to communicate with each other. An object-oriented approach provides the best programming environment for implementing the above described tasks.

The intuitive appeal of object-oriented programming is superior simulation capability of real world problems. The traditional procedural programming mold a problem into a computer, while the object-oriented programming tries to simulate functional structures of the real world problems as closely as possible. The added capabilities in data abstraction and encapsulation, and reusable tool kits make the object-oriented programming very flexible. An object can have its own variables which give the object ability to store information and knowledge. The functions of an object include processing of information, interpretion of knowledge, and creation of new objects dynamically when ever necessary. The objects communicate with each other by "messages". When an object receives a message, it performs required tasks during which it may also ask questions of another object, and then send the results to other objects.

In an object-oriented knowledge representation, certain objects can be implemented to deal with special D-sets and propositions. When an object receives a message

at a later stage, it may do the necessary additional processing and interpretation, and then submit the required results. The results from different objects can then be integrated with relevant techniques for the chosen exploration target.

6.3 Object-Oriented Knowledge Representation

The strategy employed for knowledge representation in the object-oriented program experiment is aimed at building intelligent objects that are expected to simulate the human experts in dealing with the D-sets and propositions, and to establish compatible relations among them. An intelligent object, in this case, corresponds to a D-set or a proposition. The relations then link the objects together to form a relation network.

6.3.1 The Relation Network

The basic idea of representing relations between objects as a network originates from the recent theories on problem reduction (Slagle and Gini, 1987) and frame theory (Maida, 1987). Even though a network (Figure 6.2) appears to be very similar to an AND/OR tree, a special case of the AND/OR graph, interpretion of relations between nodes is quite different from that of an AND/OR graph (Pearl, 1987). But one can still see considerable similarity in the problem reduction steps in the network. The relation network established in this research is specifically designed to satisfy requirements of the integrated exploration problem and associated inferencing steps.

An arc between two nodes indicates an evidential relationship and it connects lower level nodes to a higher level node. The lower level nodes provide evidence to a higher level node. The higher level node represents a possible source of lower level nodes if the lower level node does not represent a possible source of noise.

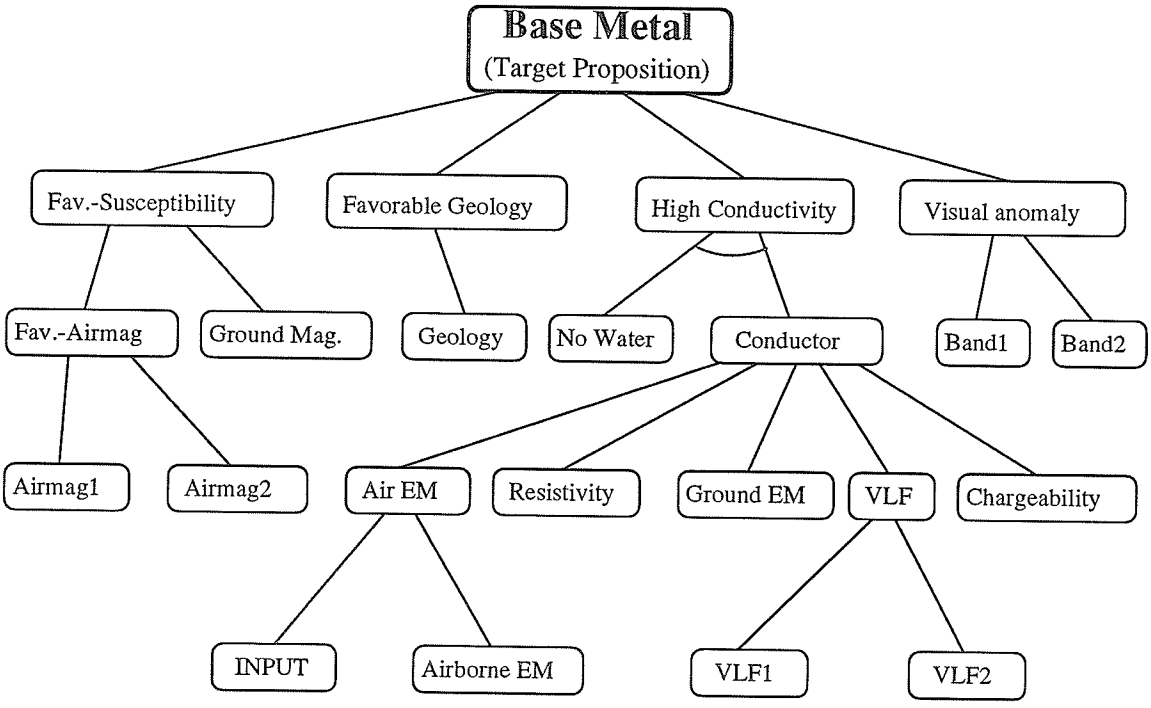

Figure 6.2: A Relation Network

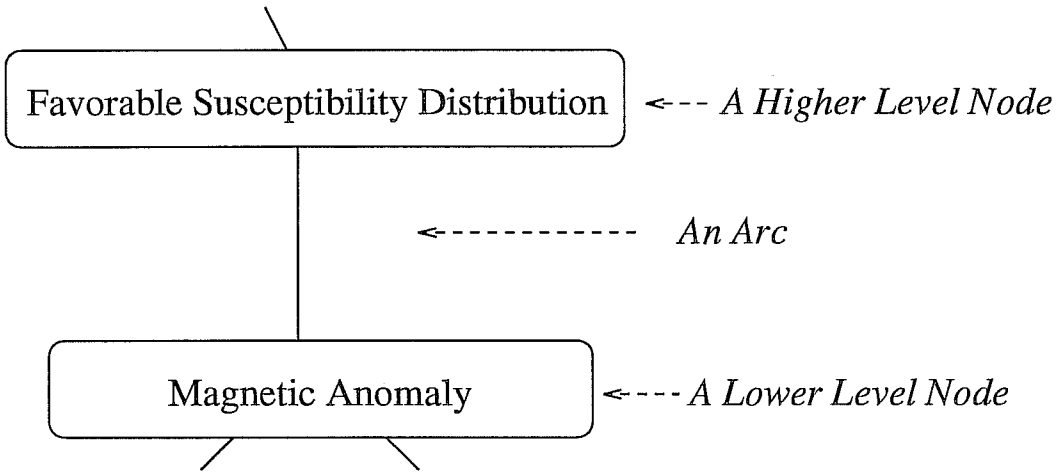

Figure 6.3: An ARC Relationship

For example, Figure 6.3 shows an arc relation between a magnetic anomaly and a favorable susceptibility distribution for a base metal deposit. This relationship can be interpreted as the identified magnetic anomaly provides evidence for a favorable susceptibility distribution. On the other hand, the susceptibility distribution may be viewed as a source of the magnetic anomaly. The situation, where the higher level node cannot be interpreted as a source of a lower level node, corresponds to the case when the lower level node represents possible noise.

The AND/OR relationship is used for the cases where more than one lower level nodes are connected to a single higher level node. An OR relationship represents a case where either one of the lower level nodes can provide evidence for the higher level node. An example is given in Figure 6.4. In this example, either the ground magnetic or the airborne magnetic anomaly can provide evidence for a high susceptibility distribution while the high level susceptibility distribution is a possible source for both ground and airborne magnetic anomalies.

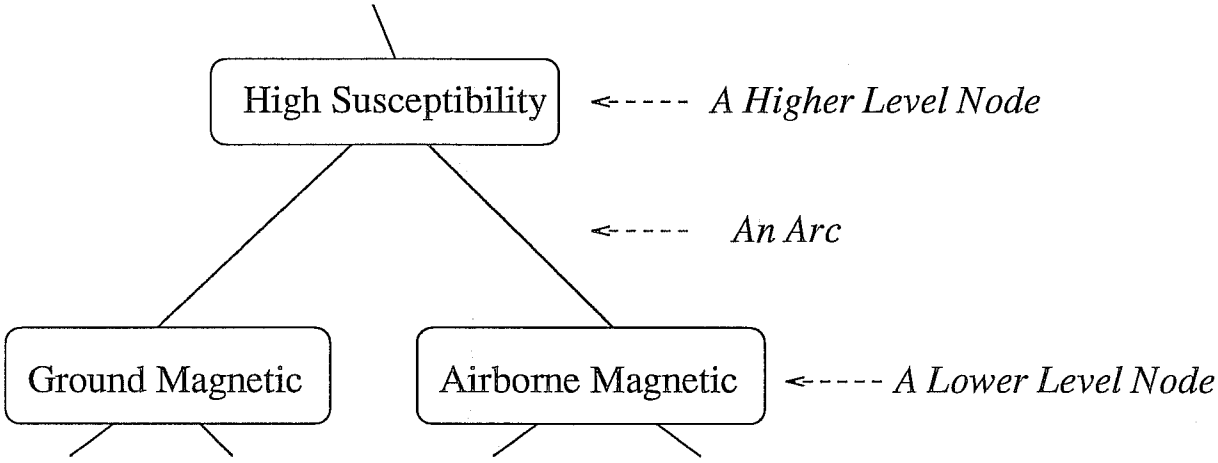

Figure 6.4: The OR Relationship

The AND relationship represents the cases where only all lower level node together can provide evidence for their higher level node. An example is given in Figure 6.5 where the arc connecting two lower level nodes indicates the AND relationship between them. In this example, existence of an EM anomaly and absence of water are simultaneously required to provide evidence to their higher level node which represents the proposition that there is a conductor. In this case, the presence of water provides a possible source of noise upon EM anomalies.

The present system allows combination of both AND and OR relationships although there is no such case in the relation network. But it does not allow a node to have more than one higher level node. If a node requires more than one upper level node, as often utilized in the problem reduction theory (Winston, 1990), it can easily be split into two or more nodes. But this may lead to difficulties in subsequent processing if the evidence represented by the split nodes cannot be treated as evidentially independent. Therefore, the objects can be connected using the above three basic relationships to form a relation network as shown in Figure 6.2.

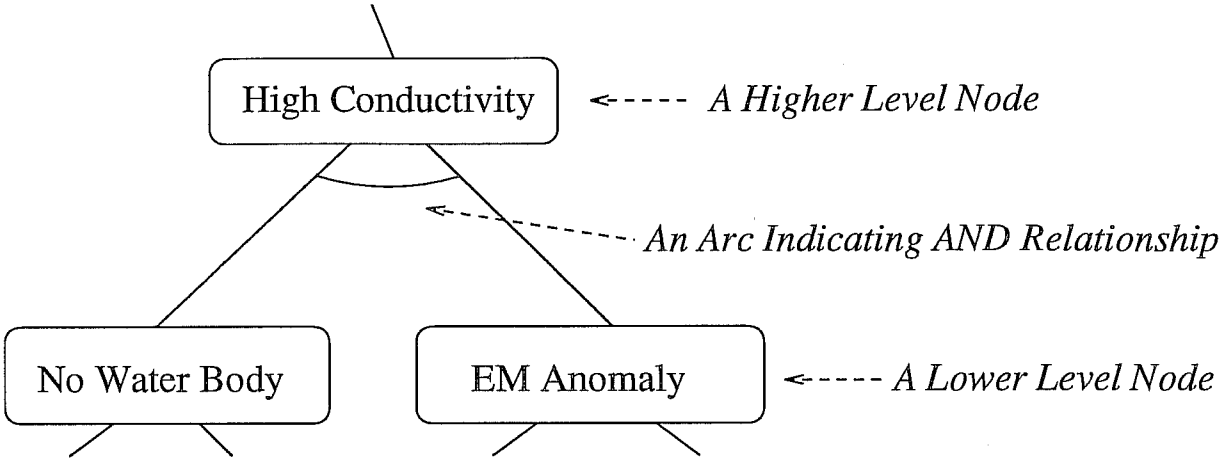

Figure 6.5: The AND Relationship

6.3.2 The Intelligent Objects

An intelligent object, as referred to in this research, is an relatively independent block of the computer program which has all the fundamental capabilities of an object: storage, processing, communication, and intelligent handling of a D-set or proposition. The knowledge of objects in the current system is represented as rules. They are divided into sub-rule bases and incorporated into corresponding objects. According to their relative locations in a network, objects can be classified as terminal objects (nodes), middle objects, and the top object.

The terminal nodes are those which have no child nodes. Examples of terminal nodes are VLF1 and Airmag1 in Figure 6.2. A terminal node accepts a D-set and acquires necessary information about the D-set from users. Each D-set must have a corresponding terminal node so that it can be integrated by the system. A terminal node has a set of functions which may be used for processing, a sub-rulebase which controls the processing and interpretation, and a simple interpreter (Figure 6.6).

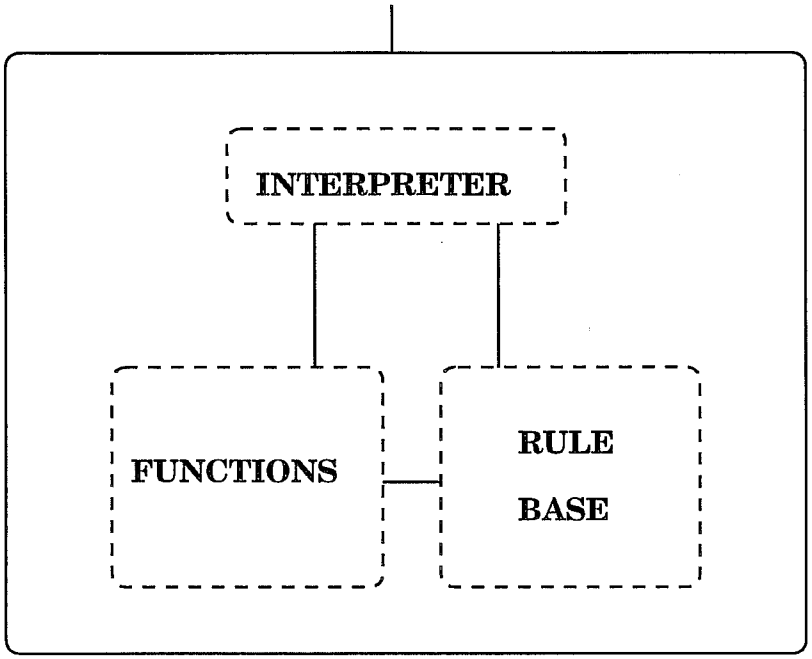

Figure 6.6: Block Diagram of a Terminal Node

The primary function of a terminal node includes receiving messages from its upper level node, searching for the corresponding D-set, acquiring necessary information about the D-set, interpreting the D-set and then, submitting a belief function about the proposition represented by its upper node. The complete procedure can either be very simple or fairly complex.

The middle nodes are those which have links with both upper and lower level nodes. Examples are Fav-Airmag, High-conductivity and VLF in Figure 6.2. A middle node consists of a sub-rulebase and an interpreter. The functional requirement of a middle node includes receiving messages from the upper node, sending messages to all its lower nodes, combining the belief functions from all lower level nodes, and calculating and submitting the belief function about the proposition represented by its upper node. Clearly, the middle nodes correspond to the propositions (concepts) or intermediate results.

The top object is the one which has no higher node. It corresponds to the explo-

ration target. There is only one target proposition, "there is a base metal deposit" in the present system.

Each terminal node or each middle node has a set of rules which is incorporated into the corresponding object. The rules built into the terminal node control the interpretation and processing of the data, and the rules in each middle node govern propagation of belief functions through it. Incorporating the rules into each corresponding object instead of using a global rule base in general improves efficiency of computation. The interpreter of an object only needs to search the rules in its own sub-rulebase. Another advantage of the present approach includes easier development and updating of the rule base.

The strategic reasons for using this type of knowledge representation structure are as follows. The knowledge required for integrated exploration and reasoning may be partitioned into two parts in an object-oriented knowledge representation. The first is the necessary knowledge required to interpret data and to manipulate the propositions. The knowledge in this category can be further partitioned into pieces of knowledge which correspond to different data sets and propositions. These pieces of knowledge can then be incorporated into objects to handle corresponding data sets and propositions to make the objects intelligent. Additional reasons for building objects are described as follow:

- Each exploration survey collects independent data sets which are interpreted for special key parameters crucial to the overall exploration strategy. The fact that knowledge needed for interpretation of one data set is different from that required for interpretation of others, requires development of separate objects adequate for handling data sets or propositions in a chosen sub-discipline.
- Certain data sets should be interpreted separately in order to keep the results of interpretation independent for subsequent processing. If a data set is interpreted

in reference to another data set, result of the interpretation will most likely depend on the referred data set and its interpretation. This may cause difficulty in subsequent processing.

The second category of the knowledge includes relations relevant to the specific exploration target with respect to the objects. These relations can be represented by a network that links the objects together. The present system allows it to be represented as an AND/OR tree, which can easily be expanded to allow more general networks.

6.4 Rules in the system

Belief functions are used to quantify both the information content and the uncertainties associated with each rule. The rules in an object have a format

If *premise*

Then *conclusion* (Bel_r),

where Bel_r is a belief function associated to the rule. The propositions of Bel_r are R_1, R_2 and R_3. R_1 presents "The rule is true"; $R_2 = \overline{R}_1$; and $R_3 = \{R_1, R_2\}$. The basic probability numbers are $m(R_1)$ which is the lower probability or degree to which the rule is true, $m(R_2)$ which is the probability of negation of R_1 or degree to which the rule is false, and $m(R_3)$ which presents the uncertainty or degree to which one does not know whether the rule is true or false. Only two independent numbers, however, are required because a belief function must satisfy

$$m(R_1) + m(R_2) + m(R_3) = 1.$$

As described earlier, the system must be able to handle two kinds of data formats; symbolic and numerical. Examples of symbolic data include lakes, basaltic intrusive,

high ground EM anomaly, etc. Examples of numerical data include a 20 *mgal* gravity anomaly and a 3500 (γ) magnetic anomaly. Three logic words are implemented to deal with the numerical data. They are *less-than*, *between* and *greater-than*. These algebraic symbols pre-classify numeric data into a manageable number of categories. Below are two examples of the rules in the objects; one is for numerical data and the other is for symbolic data.

Example–1 **If** the *magnetic anomaly* is *greater-than* 3500 γ,
then there is a *high-magnetic* anomaly (0.5 0.05 0.45).

Example–2 **If** there exist *mafic-intermediate-volcanic* rocks,
then there is a *favorable geological* environment for a base metal deposit (0.5 0.1 0.4).

6.5 The Inference Engine

The inference engine is also divided into two parts to suit the corresponding knowledge representation structure. The first part is made of a relatively simple interpreter and can be incorporated into the objects. When it receives a message or is activated, it searches applicable rules in its sub-rule-base and applies the rules to carry out the tasks which may include processing, interpretation, and evidence combination. During this step the object may communicate with other objects. Results of the task are belief functions to be submitted to its upper node. The second part involves a systematic backward search over the relation network to find applicable objects and activate them, and then, forward chains to propagate the belief functions to the top node.

6.5.1 Search Algorithm

The search mechanism in an AI system is to find a solution path from an initial node to the goal node (Winston, 1990). The algorithm developed here searches systematically through the relation network (Figure 6.2) and finds all existing evidence relevant to the top proposition, activates pertinent objects, and propagates the belief functions to the top proposition. The systematic search method used in this system is depth-first and it is as follows:

1 Put the top node on a list, called OPEN, of unvisited nodes. If the top node has no lower node, exit with failure.

2 If OPEN is empty, exit successfully.

3 If the last node, n, of OPEN is a terminal node or has no unvisited successor, do the following:

(1) activate n;

(2) set n visited;

(3) move n from OPEN.

4 If n has an unvisited successor, get a successor of n and place it last in OPEN.

5 Go to step 2.

6.5.2 Inference Mechanisms

Inference mechanisms are the ways in which belief functions are propagated through the network. In this research, Dempster-Shafer evidential belief function theory

(Dempster, 1968; Shafer, 1976) is used as the basis for propagating belief functions through the relation network. Three basic types of inference mechanisms corresponding to the knowledge representation structure are used in this research. They are called OR, AND, and PASS-RULE operators.

Suppose we have two belief functions Bel_1 and Bel_2, each of which can represent either a D-set or a proposition. The propositions of Bel_1 are H_1, H_2 and H_3, where H_1 represents "the proposition is true", $H_2 = \overline{H}_1$, and $H_3 = \{H_1, H_2\}$. The corresponding basic probability measures are

$$\begin{aligned} m_1(H_1) &= b_1 \\ m_1(H_2) &= d_1 \\ m_1(H_3) &= u_1 \end{aligned}$$

and they correspond to belief, disbelief and uncertainty about the proposition. Similarly, the propositions of Bel_2 are S_1, S_2 and S_3, where S_1 is "the proposition is true", $S_2 = \overline{S}_1$, and $S_3 = \{S_1, S_2, \}$. The corresponding basic probability numbers are

$$\begin{aligned} m_2(S_1) &= b_2 \\ m_2(S_2) &= d_2 \\ m_2(S_3) &= u_2. \end{aligned}$$

Let the propositions of Bel which we are trying to establish by combining Bel_1 and Bel_2 be

$$\{T_o, \overline{T}_o, T_{\Theta_o}\}$$

where T_o represents "the proposition T is true from Bel_1 or Bel_2", $\overline{T}_o$ is its negation and $T_{\Theta_o} = \{T_o, \overline{T}_o\}$. According to the Dempster's rule of combination, we have

$$\sum_{\substack{i,j \\ H_i \cap S_j = T_o}} m_1(H_i) m_2(S_j) = b_1 b_2 + u_1 b_2 + u_2 b_1$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \overline{T}_o}} m_1(H_i) m_2(S_j) = d_1 d_2 + u_1 d_2 + u_2 d_1$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \Theta}} m_1(H_i) m_2(S_j) = u_1 u_2$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i) m_2(S_j) = b_1 d_2 + b_2 d_1$$

and we have probability measures for Bel from the OR operator

$$\begin{aligned} m(T_o) &= \frac{b_1 b_2 + b_2 u_1 + b_1 u_2}{1 - b_1 d_2 - b_2 d_1} \\ m(\overline{T}_o) &= \frac{d_1 d_2 + d_1 u_2 + d_2 u_1}{1 - b_1 d_2 - b_2 d_1} \\ m(T_{\Theta_o}) &= \frac{u_1 u_2}{1 - b_1 d_2 - b_2 d_1}. \end{aligned} \tag{6.1}$$

This formula is the same as (5.17).

Now let the propositions of Bel from the AND operator by combining Bel_1 and Bel_2 be

$$\{T_a, \overline{T}_a, T_{\Theta_a}\}$$

where T_a represents "the proposition T is true from Bel_1 *and* Bel_2", $\overline{T}_a$ is its negation, and $T_{\Theta_a} = \{T_a, \overline{T}_a\}$. According to the Dempster's rule of combination, we have

$$\sum_{\substack{i,j \\ H_i \cap S_j = T_a}} m_1(H_i) m_2(S_j) = b_1 b_2$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \overline{T}_a}} m_1(H_i) m_2(S_j) = d_1 d_2$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \Theta}} m_1(H_i) m_2(S_j) = u_1 u_2 + b_1 u_2 + b_2 u_1 + d_1 u_2 + d_2 u_1$$

$$\sum_{\substack{i,j \\ H_i \cap S_j = \emptyset}} m_1(H_i) m_2(S_j) = b_1 d_2 + b_2 d_1$$

and we have probability measures for Bel from the AND operator

$$\begin{aligned} m(T_a) &= \frac{b_1 b_2}{1 - b_1 d_2 - b_2 d_1} \\ m(\overline{T}_a) &= \frac{d_1 d_2}{1 - b_1 d_2 - b_2 d_1} \\ m(T_{\Theta_a}) &= \frac{u_1 u_2 + b_2 u_1 + b_1 u_2 + d_1 u_2 + d_2 u_1}{1 - b_1 d_2 - b_2 d_1}. \end{aligned} \tag{6.2}$$

Let Bel_r be a belief function associated to a rule. The IF part (condition) of the rule represents the evidence and the THEN part (conclusion) represents a hypothesis. Three propositions of the belief function representing the uncertainties associated with the rule are R_1, R_2 and R_3, where R_1 represents "the rule is true", $R_2 = \overline{R}_1$, and $R_3 = \{R_1, R_2\}$, respectively. The probability numbers for the given propositions are

$$\begin{aligned} m_r(R_1) &= r_1 \\ m_r(R_2) &= r_2 \\ m_r(R_3) &= r_3. \end{aligned}$$

Let the belief function Bel_e represent the condition of the rule and the propositions be S_1, S_2 and S_3, where S_1 is "the condition is true", $S_2 = \overline{S}_1$, and $S_3 = \{S_1, S_2\}$. The corresponding probability numbers are

$$m_e(S_1) = s_1$$
$$m_e(S_2) = s_2$$
$$m_e(S_3) = s_3.$$

Let Bel_h be a belief function to be established and the propositions be H_1, H_2 and H_3, where H_1 is "the conclusion of the rule is true", $H_2 = \overline{H}_1$, and $H_3 = \{H_1, H_2\}$. According to the Dempster's rule of combination,

$$\sum_{\substack{i,j \\ S_i \cap R_j = H_1}} m_e(S_i) m_r(R_j) = s_1 r_1$$

$$\sum_{\substack{i,j \\ S_i \cap R_j = H_2}} m_e(S_i) m_r(R_j) = s_1 r_2 + s_2 r_1$$

$$\sum_{\substack{i,j \\ S_i \cap R_j = H_3}} m_e(S_i) m_r(R_j) = s_1 r_3 + s_3 r_1 + s_3 r_2 + s_2 r_3 + s_3 r_3$$

$$\sum_{\substack{i,j \\ S_i \cap R_j = \emptyset}} m_e(S_i) m_r(R_j) = s_2 r_2$$

and then the probability measures for Bel_h from the PASS-RULE operator become

$$m(H_1) = \frac{s_1 r_1}{1 - s_2 r_2}$$

$$m(H_2) = \frac{s_1 r_2 + s_2 r_1}{1 - s_2 r_2} \tag{6.3}$$

$$m(H_3) = \frac{s_1 r_3 + s_3 r_1 + s_3 r_2 + s_2 r_3 + s_3 r_3}{1 - s_2 r_2}.$$

The PASS-RULE mechanism simulates the inference procedure from the evidence to a hypothesis or proposition. It functionally transforms the belief function representing the condition of a rule into a new belief function which represents the conclusion of the rule. This transform is controlled by the belief functions associated with the rule. These three mechanisms are used in the system to propagate the belief functions through the relation network.

6.6 Problems with Dependent Evidence

Dependent evidence and concept management have always been problematic in multiple data set integration. Bayesian probability theory has a clear definition of the statistical independence. There are methods to calculate the probabilities associated with mutually dependent events but they usually introduces great complexity in solving real world problems. Assumptions, such as conditional independence (Chung and Moon, 1990), sometimes have to be made to apply the theory to real world problems, even when the evidence is clearly dependent from one's experience or physical principles. In certain situations, assumptions such as conditional independence or local independence can lead to unacceptable conclusions (Konolige, 1979). Actually, the traditional probability theory can not adequately handle the problem of dependent evidence and it requires more solid theoretical basis and theoretical approaches that are easy to implement.

The belief function theory has been recently generalized from the traditional prob-

ability theory (Dempster, 1967, 1987) and can be regarded as a way of representing inferences from evidence within the probabilistic framework (Watson, 1987). Some difficulty persists for belief function theory to have a clear definition on the detailed meanings of independence. In the application of the Dempster's rule to determine a new belief function for a chosen hypothesis on the basis of two or more pieces of evidence, the premise is a rather vague notion that the pieces of evidence should be independent in some way or from the distinct bodies of evidence.

Since the belief function theory includes the Bayesian theory as a special case (Shafer, 1976), pieces of evidence that are independent in the belief function framework are not necessarily statistically independent in a Bayesian probability model. Evidential independence (Moon, 1990) is used in this research to differentiate the concept of independence in a belief function system from the concept of statistical independence in the Bayesian probability models.

The belief function theory depends on use of information (data) as evidence for a chosen hypothesis or proposition. Whether pieces of evidence is evidentially independent or not relies often on the hypothesis to be proven. For example, let us consider a case where two airborne magnetic surveys were carried out over the same location but at different periods. If the two measurements are used to prove a hypothesis that there is a base metal deposit at a specific location, they are evidentially dependent. If the same measurements are used to prove a hypothesis that there is an airborne magnetic anomaly at the same location, they can be interpreted as evidentially independent. This means that evidential independence is relative to the chosen frame of discernment. Using this unique property of the knowledge-based approach, a knowledge representation structure can be organized in such a way that the lower level nodes are evidentially independent relative to their higher level nodes. This approach is actually related to the fundamental concepts of reframing (Shafer, 1984) and partitioning (Shafer and Logan, 1985).

Shafer (1984) remarked that independence is always relative to the frames of discernment. By explicitly introducing significant common uncertainties into a frame of discernment, remaining uncertainties may be treated as independent with respect to that frame of discernment. Shafer and Logan (1985) introduced the idea of partitioning a frame of discernment. Such a partition can itself be regarded as a frame of discernment. By partitioning the frame of discernment, the complexity of the Dempster's rule can be greatly reduced. These ideas are implemented in the knowledge representation structure and in construction of new belief functions. If a complete target-oriented exploration process can be represented as a frame of discernment, it will include all the evidence (data sets), propositions, and rules. In this frame of discernment, many propositions are brought explicitly into the frame, such as "there is a high conductivity", "there is a high magnetic anomaly", etc. The rules in the frame connect evidence and propositions together and functionally control the propagation of belief functions from evidence to a proposition and from a proposition to another proposition. If a frame only has one proposition that represents a specific exploration target, and if it has only an OR relation, it becomes the same as the case described in Chapter 5 of this thesis and in Moon (1990). In the test example with real data in Chapter 5, only one of the two D-sets (airborne magnetic surveys) in the Snow Lake area is used. But both can be naturally used in this knowledge-based system.

6.7 Some Comments on the Prototype System

To use the prototype system in a research or exploration project, several support files have to be established. These files include legend files for different data layers, a file to tell the system where to search for data, rule base, and the legend files. The format of these files are given in Appendix A. In addition, users have to make judgements based on the relative precision of different D-sets and to provide a probability

number between 0 to 1 for each D-set, where 0 means that nothing is believable in the corresponding D-set and 1 means that the information content of the D-set is completely accurate. The system will construct a belief function according to the given probability number. The propositions for the belief function are H_1, H_2, and H_3, where H_1 represent "the information represented by the D-set is true", $H_2 = \overline{H}_1$, and $H_3 = \{H_1, H_2\}$. Suppose that p is a probability number given by a user such that $0 \leq p \leq 1$, the probability measures assigned by the system are

$$\begin{aligned} m(H_1) &= p \\ m(H_2) &= 0 \\ m(H_3) &= 1 - p. \end{aligned}$$

This belief function is used for the whole D-set and the present system cannot process probability values that vary with location and for different data descriptions in the same D-set.

The system can also be customized easily for different geological regions or exploration projects by modifying the relation network and/or rule base. The formats of these files are also described in Appendix A.

As described earlier, the system can only combine the D-sets which have corresponding terminal objects and cannot create objects dynamically. Although it is not too difficult to implement new objects, one has to be familiar with the system before actually doing it.

The interpreters incorporated in the objects are very simple. The system may be expanded to make it more flexible and capable of handling more difficult problems.

6.8 Test of the Prototype System

The prototype system is tested using D-sets from both the Farley Lake area (Figure 2.1) and Snow Lake area (Figure 2.2). The outputs of the system are belief function maps which have the same meanings as those described in Chapter 5. Figure 6.7 displays the spatial distribution of support for a base metal deposit in Farley Lake area. Similarly to the support map in Chapter 5, the higher support is found in the west central area. Figure 6.8 shows the disbelief for the exploration target. The disbelief is low in most of the test area and higher disbelief is found in the areas where there is more efficient data coverage and no significant anomalies detected. The low uncertainties (Figure 6.9) are obtained in the middle west areas where there are more efficient data coverage. The high level of uncertainty is visible in the areas with less data and/or less efficient data. High level of uncertainty with lower disbelief generally suggests that more data should be acquired in the area to reach a decision. The plausibility (Figure 6.10) is the sum of belief and uncertainty. It can be interpreted in conjunction with other output maps.

An apparent feature of the results of the Snow Lake area is that areas with high support are closely related to EM anomalies (Figure 6.11). This is because the EM data in this area are more effective for base metal exploration than other data. There are Zn-Cu deposit discoveries in the southwest part of the test area. Unfortunately, only geological and one airborne magnetic survey are available over the area with the mineral occurrences and subsequently the support values are not high. But the disbelief (Figure 6.12) is low and the uncertainty (Figure 6.14) and plausibility (Figure 6.13) are high. This fact indicates that more information is needed to make a decision.

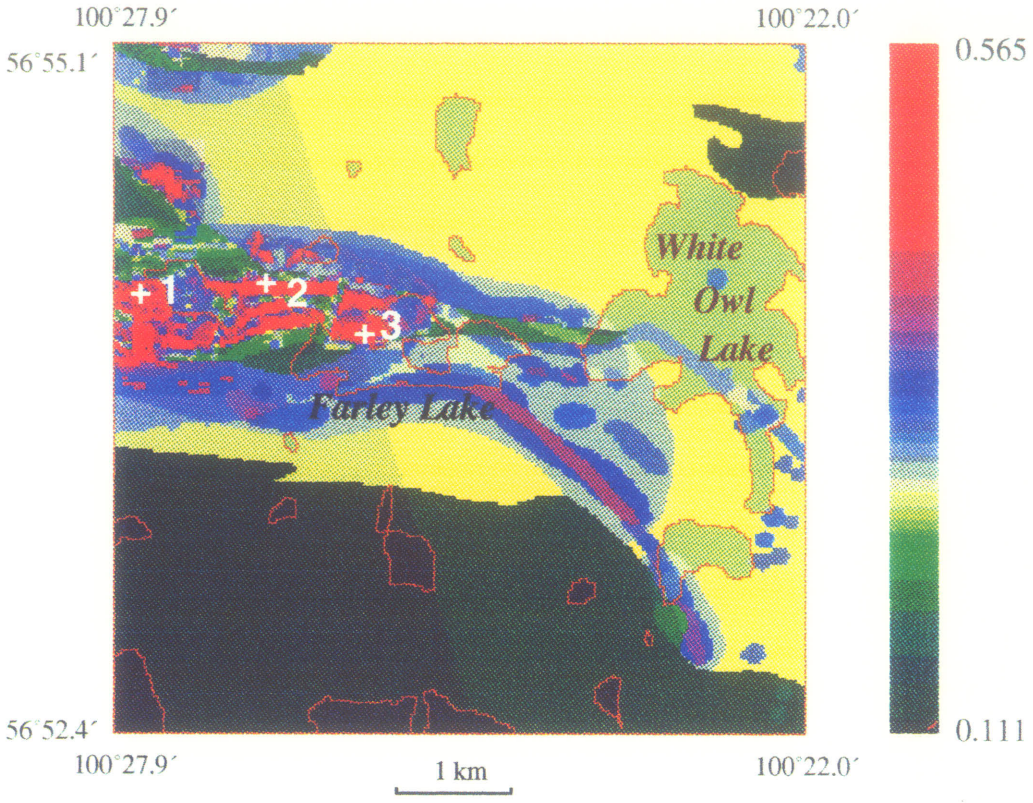

Figure 6.7: Support Map for a Base Metal Deposit (Farley Lake)
1 – Iron; 2, 3 – Gold.

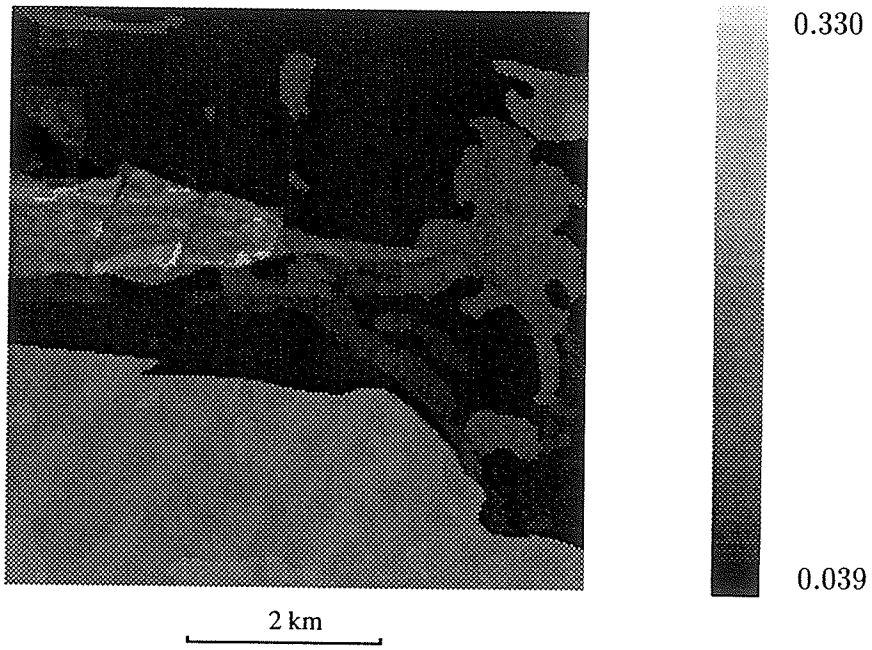

Figure 6.8: Disbelief Map for a Base Metal Deposit (Farley Lake)

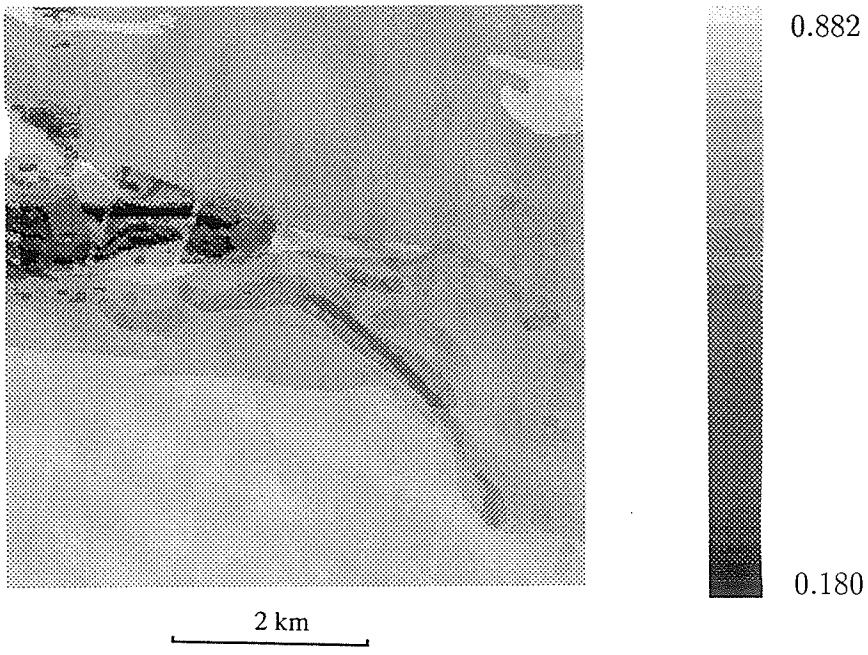

Figure 6.9: Uncertainty Map for a Base Metal Deposit (Farley Lake)

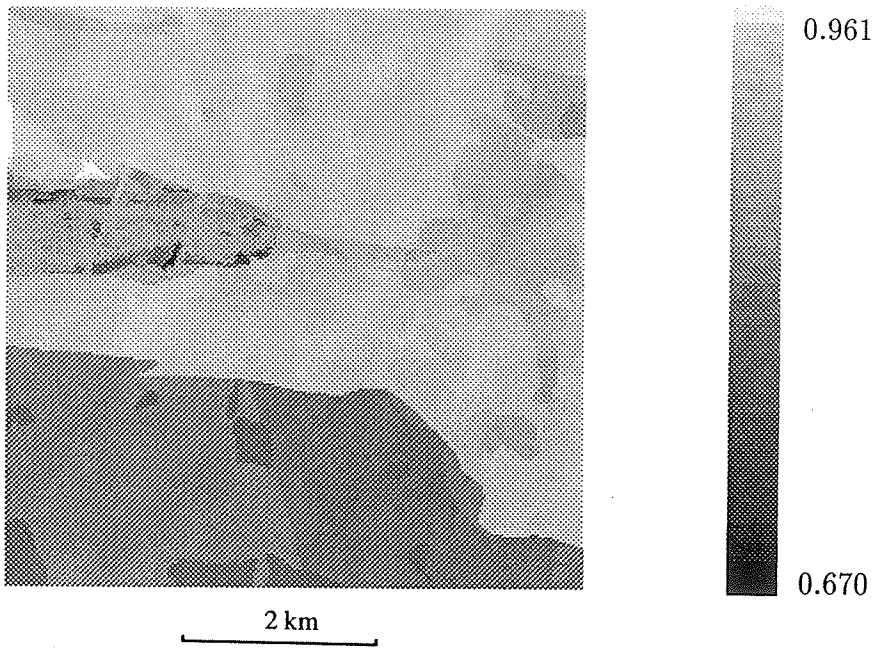

Figure 6.10: Plausibility Map for a Base Metal Deposit (Farley Lake)

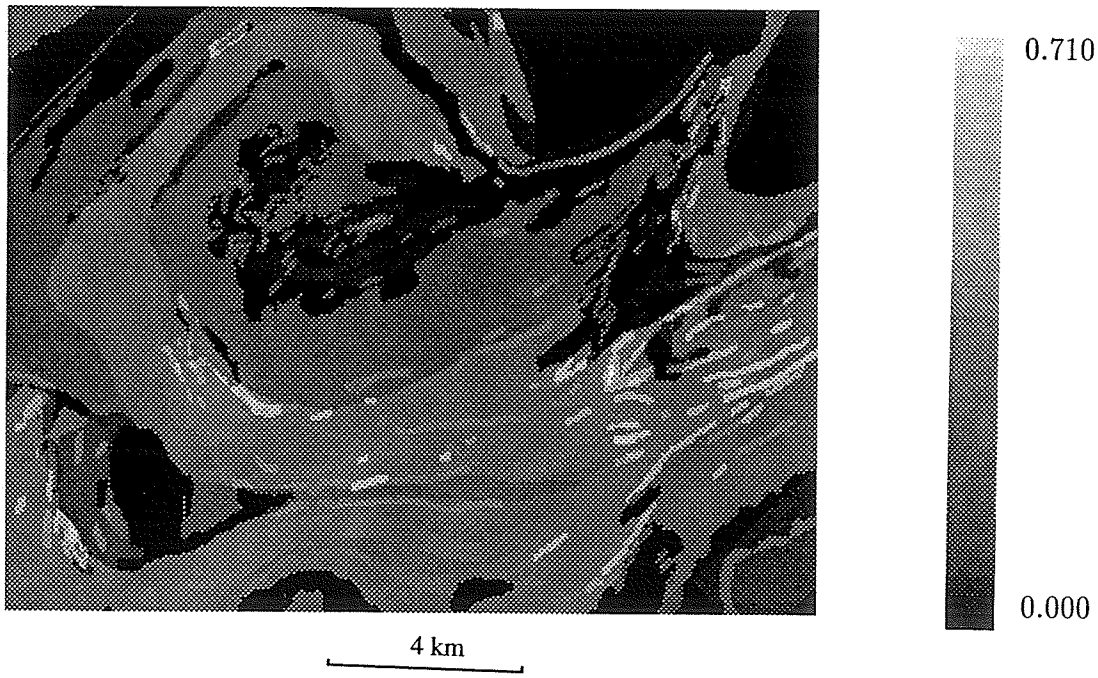

Figure 6.11: Support Map for a Base Metal Deposit (Snow Lake)

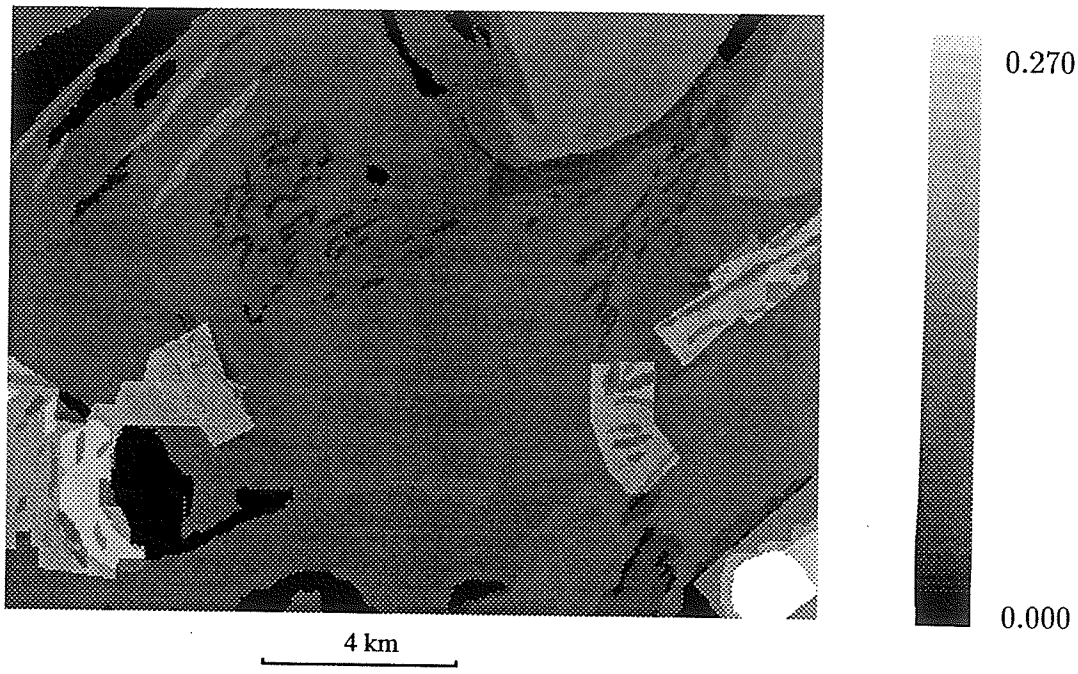

Figure 6.12: Disbelief Map for a Base Metal Deposit (Snow Lake)

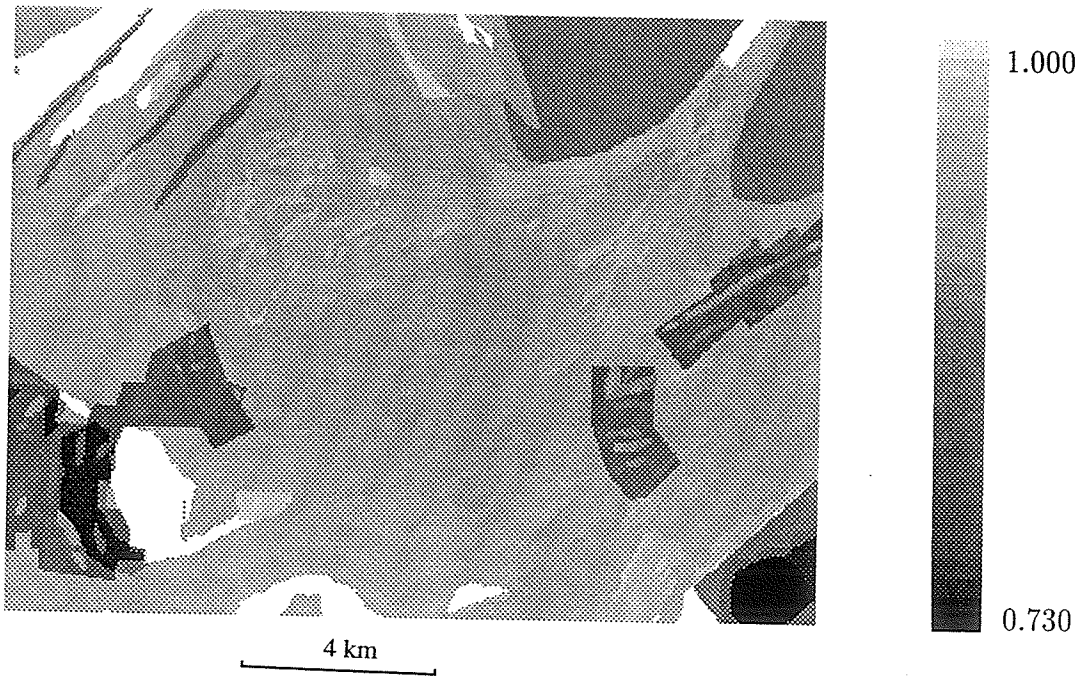

Figure 6.13: Plausibility Map for a Base Metal Deposit (Snow Lake)

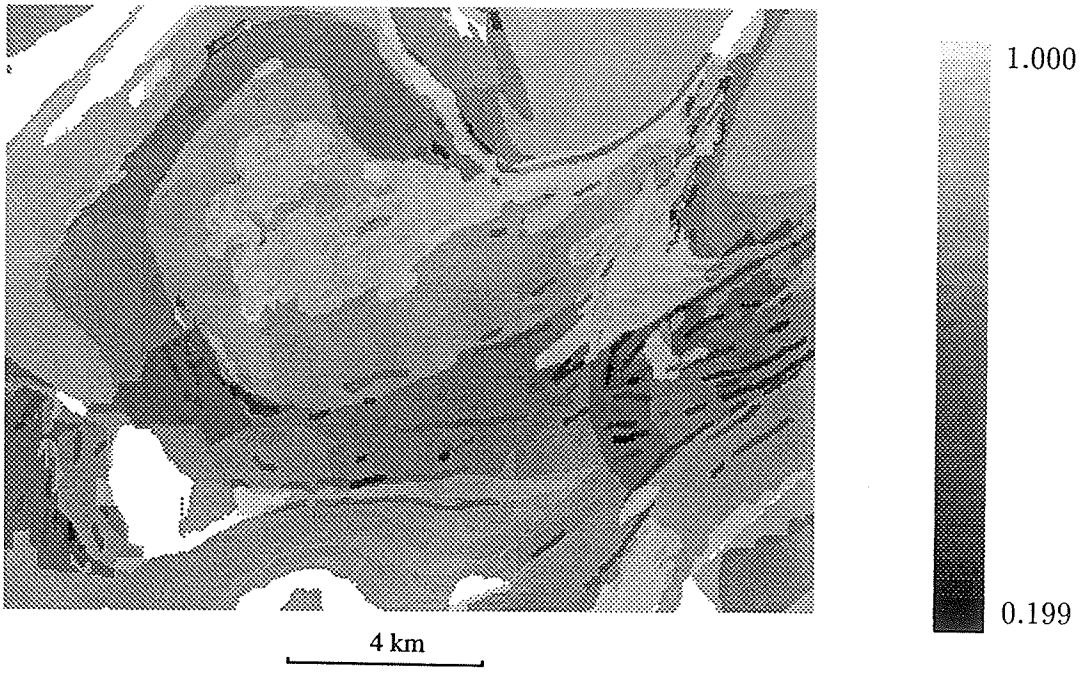

Figure 6.14: Uncertainty Map for a Base Metal Deposit (Snow Lake)

6.9 Conclusions on the Prototype Development

Use of evidential belief functions in knowledge representation has provided a theoretical basis for the current spatial reasoning system. It allows uncertain information and knowledge to be represented in a more precise natural manner. Subsequent object-oriented knowledge representation structure partitions the complicated tasks of integrated exploration into sub-tasks which are more manageable by using artificial intelligence techniques. The dependent evidence can be better utilized by introducing important uncertainties explicitly and by organizing the relation network properly. The test example has successfully outlined the favorable exploration target areas. The knowledge representation structure described above appears to be a promising approach for integrated exploration. The inference mechanisms proposed for uncertainty propagation have worked well in the test examples. These mechanisms can be

used to simulate human reasoning processes.

Chapter 7

Conclusion

This thesis research has developed a set-theoretic representation for geological and geophysical information integration. The theoretical basis of the fuzzy set approach has been established and tested using real exploration data sets. The Dempster-Shafer evidential belief function approach has been studied systematically and also tested using the same data sets. An object-oriented and map-based prototype expert system was then developed to utilize the knowledge based approach for this complicated integrated exploration problem.

Integration of geological and geophysical data for resource exploration can be represented as G-mappings based on a data structure, the X-set. Integrated exploration problems can be further studied based on this mathematical representation approach. Based on this set-theoretic representation, exploration activities can generally be classified into two categories. The first includes data acquisition and improvement. The backbone of this category is basically the classical set theory. The second includes integration and interpretation of the data to search the exploration targets we are interested in. Tasks in this category often cannot be represented precisely in the framework of a classical set theory. Fuzzy set and probabilistic approaches are usually studied for this category of exploration tasks.

Fuzzy set theory provides a tool that can adequately represent imprecise and sometimes non-linear geoscience information. The data represented in the fuzzy space can then be processed and integrated in fuzzy space more precisely and effectively than in the framework of classical set theory.

Dempster-Shafer evidential belief function theory is a generalized form of the traditional probability theory. It provides a more natural theoretical basis for representing geological and geophysical information. It does not specifically require the remainder of a probability to be committed to its negation. This property allows one to distinguish between lack of evidence and negative evidence. The geological and geophysical data represented as belief functions can then be combined using Dempster's rule of combination. Consequently, the results provide a more realistic description about the exploration target or the target proposition. The degree of conflict utilized in this research provides a very useful quantity for making a final decision. Both fuzzy set and Dempster-Shafer approaches can be used in the areas where there are none or only very few discovered exploration targets. But some ambiguity and difficulty still exist in mapping geological and geophysical information into the fuzzy space or probability space.

The prototype expert system developed in this research demonstrates that a knowledge-based system can be an effective tool to overcome some of the above mentioned difficulties. The object-oriented knowledge representation decomposes the integrated resource exploration problem into subproblems which are more manageable by AI approaches. Furthermore, the problem of dependent information integration can generally be better dealt with in the knowledge-based systems of this type. The prototype expert system can be customized for different mineral types by modifying the relation network and the rule bases without concerning about other parts of the system. It will be more useful for exploration industry if the system can be integrated with a good GIS package and take advantage of its data base management and

display facilities.

Although the above approaches have been tested using real exploration data sets from two exploration areas, more tests in different geological regions and for different mineral deposit types are needed. Incorporating these approaches in a good GIS system or a mineral exploration software is also an important step for industry application. The prototype expert system can be refined and expanded for wider and more practical applications. Further testing of the system will certainly be necessary in further upgrading of the system. Furthermore, a detailed and comparative study of the fuzzy set, belief function, Bayesian, regression, and different knowledge-based models in the same or similar areas would be helpful for a more in-depth understanding of these approaches.

Bibliography

[1] Agterberg, F. P., Bonham-Carter, G. F., and Wright, D. F., 1990. Statistical pattern integration for mineral exploration. Gaaál, G., and Merriam, D. F., eds., Computer Applications in Resource Estimation, Pengamon Press, pp. 1-21.

[2] Agterberg, F. P., 1989. Computer programs for mineral exploration. Science, Vol. 245, pp. 76-81.

[3] Aminzadeh, F., 1991, Where are we now and where are we going? In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 3-32.

[4] An, P., Moon, W. M., and Bonham-Carter, G. F., 1992. On knowledge-based approach of integrating remote sensing, geophysical and geological information. Proceedings of IGARSS'92, IEEE.

[5] An, P., Moon, W. M., and Rencz, A., 1991. Integration of geological, geophysical, and remote sensing data using fuzzy set theory. Canadian Journal of Exploration Geophysics, Vol. 27, No. 1., pp. 1-11.

[6] An, P., 1989. Application of AI techniques to geophysical data integration and mineral exploration. Ph.D. thesis proposal, University of Manitoba, Canada.

[7] Argialas, D. P., 1990. Computational image interpretation models: an overview and a perspective. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 871-886.

[8] Bailes, A. H. and Syme, E. C., 1989. Geology of the Flin Flon-White Lake Area. Energy and Mines, Provencial Government of Manitoba, Geological Report GR87-1.

[9] Bailes, A. H. and Galley, A. G., 1991. Geological setting of base metal mineralization in the Anderson Lake area. Manitoba Energy and Mines, Report of Activities 1991. pp. 8-13.

[10] Bamburak, J. D., 1990. Metallic mines and mineral deposits of Manitoba. Energy and Mines, Provencial Government of Manitoba, Canada. Open file report OF90-2.

[11] Barr, A. and Feigenbaun, E. A., eds., 1981. The handbook of artificial intelligence. William Kaufmann Inc.

[12] Blonda, P., Polosa, R. L., Losito, S. A., Pasquariello, M. G., Posa, F., and Ragno, D., 1989. Classification of multi-temporal remotely sensed images based on a fuzzy logic technique. Proceedings of IGARSS'89, IEEE 89CH2768-0, pp. 834-837.

[13] Bonham-Carter, G. F., Agterberg, F. P., and Wright, D. F., 1989. Weights of evidence modelling: a new approach to mapping mineral potential. In Statistical Applications in the Earth Sciences, Agterberg, F. P. and Bonham-Carter, G. F. (eds), Geological Survey of Canada, Paper 89-9, pp. 171-183.

[14] Bonham-Carter, G. F., Agterberg, F. P., and Wright, D. F., 1988. Integration of geological data for gold exploration in Nova Scotia. Photogrammetric Engineering and Remote Sensing, Vol. 54, No. 11, pp. 1585-1592.

[15] Bonnet, A. and Dahan, C., 1983. Oil-well data interpretation using expert system and pattern recognition technique. Proceedings IJCAI-83 pp: 185-189.

[16] Campbell, A. N., Hollister, V. F., Duda, R. O., and Hart, P.E., 1982. Recognition of a hidden mineral deposit by an artificial intelligence program. Science, v. 217, no. 4563, pp. 927-929.

[17] Campbell, F. H. A., 1969. Turnbull Lake area. in Summary of Geological Feildwork, Department of Energy and Mines, Provencial Government of Manitoba, Canada, Geological Paper 4/69, pp. 15-20.

[18] Campbell, W. J. and Cromp, R. F., 1990 Evolution of an intelligent information fusion system. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 867-870.

[19] Chevillard, R., and Genaile, B., 1970. Ground horizontal loop EM survey map. Department of Energy and Mines, Provincial Government of Manitoba, Canada, Assessment file 91436.

[20] Chung, C. F., and Moon, W. M., 1990. Combination rules of spatial geoscience data for mineral exploration. Extended abstract, ISME-AI'90 (Tokyo), pp. 131-141.

[21] Clark, G. S., 1980. Rubidium-strontium geochronology in the Lynn Lake greenstone belt, Northwestern Manitoba. Manitoba Mineral resources Division, Geological Paper GP80-2.

[22] Conrad, M. A. and Beightol, D. S., 1988. Expert systems identify fossils and manage large paleontological database. Geobyte, February.

[23] Coppens, F., 1991. A rule-based system for the determination of seismic velocities. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 33-57.

[24] Dempster, A. P., 1966. New methods for reasoning towards posterior distributions based on sample data. Annals of Mathematical Statistics, Vol. 37, pp. 355-374.

[25] Dempster, A. P., 1967a. Upper and lower probabilities induced by a multivalued mapping. Annals of Mathematical Statistics, Vol. 38, pp. 325-339.

[26] Dempster, A. P., 1967b. Upper and lower probability inference based on a sample from a finite univariate population. Biometrika, Vol. 54, pp. 515-528.

[27] Dempster, A. P., 1968 A generalization of Bayesian inference. Journal of the Royal Statistical Soceity, series B, Vol. 30, pp. 205-247.

[28] Dempster, A. P., and Kong, A., 1987. Comment. Statistical Science, Vol. 2, No. 1, pp. 32-36.

[29] Dubois, D., and Prade, H., 1985. Combination and propagation with belief functions. IJCAI-85 Proceedings, pp. 111-113.

[30] Duda, R. O. and Reboh, R., 1984. AI and decision making: the PROSPECTOR experience. In W. Reitman (ed.) *Artificial Intelligence Applications for Business*, Norwood, N.J.: Ablex Publishing Corporation.

[31] Duda, R. O., 1980. The PROSPECTOR system for mineral exploration. (Final Report, SRI Project 8172), Menlo Park, CA: SRI International, Artificial Intelligence Center.

[32] Duda, R. O., Hart, P. E., Konolige, K. and Reboh, R., 1979. A computer-based consultant for mineral exploration. (Final Report, SRI Project 6415), Menlo Park, CA: SRI International, Artificial Intelligence Center.

[33] Dulac, J. C., 1991 An intelligent front end for interactive seismic processing. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 85-112.

[34] Egenhofer, M. J., and Frank, A. U., 1990. LOBSTER: combining AI and database techniques for GIS. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 919-926.

[35] Ernst, G. W. and Newell, A., 1969. A case study in generality and problem solving. Academic Press, New York.

[36] Fabbri, A. G., 1984. Image processing of geological data. Van Nostrand Reinhold Company, pp. 99-112.

[37] Fang, J. H., Chen, H. C. and Wright, D., 1991. A fuzzy expert system for thin-section mineral identification. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 203-220.

[38] Fedikow, M. A. F., Ostry, G., Ferreira, K. J., and Galley, A. G., 1989. Mineral deposits and occurrences in the File Lake area, NTS 64k/16. Department of Energy and Mines, Provencial Government of Manitoba, Canada. Mineral Deposit Series Report No. 5.

[39] Ferreira, K. J. and Fedikow, M. A. F., 1991. Results of a rock geochemical study in the area of rod Cu-Zn deposit, Snow Lake, Manitoba. Department of Energy and Mines, Provencial Government of Manitoba, Canada. Economic Geology Report ER91-1.

[40] Fikes, R. E. and Nilsson, N. J., 1971. STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, Vol. 2, pp. 189-208.

[41] Fowler, M. L., 1987. SRPS: A knowledge-based system for predicting source rock potential. Geobyte, November, pp. 24-30.

[42] Fox, M., 1990. General chairman's message. In Proceedings, IEEE sixth conference on artificial intelligence applications. IEEE Computer Seceity Press.

[43] Froese, E. and Moore, J. M., 1980. Metamorphism in the Snow Lake area, Manitoba. Geological Survey of Canada, Paper 78-27.

[44] Gilbert, H. P., Syme, E. C., and Zwanzig, H. V., 1980. Geology of the metavolcanic and volcaniclastic metasedimentary rocks in the Lynn Lake area. Manitoba Mineral Resources Division, Geological Paper GP-1.

[45] Goodenough, D. G., Robson, N. A., and Fung, K. B., 1989. Expert systems and environmental change-information extraction from remote sensing data. Proceedings of IGARSS'89, IEEE 89CH2768-0, pp. 823-828.

[46] Gordon, J., and Shortliffe, E. H., 1985. A method for managing evidential reasoning in hierarchical hypothesis space. Artificial intelligence, Vol. 26, pp. 323-358.

[47] Hadipriono, F. C., Lyon, J. G., Li, T. W. H., and Argialas, D. P., 1990. The Development of a knowledge-based expert system for analysis of drainage patterns. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 905-909.

[48] Hayes-Roth, F., 1987. Expert systems. In Shapiro, S. C., Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[49] Hosain, I. T., 1988. An update summary and evaluation of geophysical data from open assessment files of the Flin Flon-Snow Lake greenstone belt. Manitoba Energy and Mines, Open File Report OF87-11.

[50] INCO, 1954. Preliminary report of the airborne EM survey over Lynn Lake – Farley Lake area. Manitoba, Canada.

[51] Kahn, G. and McDermott, J., 1984. MUD system. *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society.

[52] Klemme, H. D., 1980. Petroleum basins–classifications and characteristics. Journal of Petroleum Geology, Vol. 3, No. 2, pp. 187-207.

[53] Konolige K., 1979. Bayesian methods for updating probabilities. In Duda, R. O. et al. (1979), *A Computer-based Consultant for Mineral Exploration* (Final Report, SRI Project 6415). Menlo Park, CA: SRI International, Artificial Intelligence Center.

[54] Kuo, T., and Startzman, R. A., 1987. Field-scale stratigraphic correlation using artificial intelligence. Geobyte, Vol. 2, No. 2.

[55] Kuratowski, K., and Moslowski, A., 1976 Set theory-with an introduction to descriptive set theory. North-Holland.

[56] Lafue, G. M. E., Pajot, D. and Winchester, A. J., 1991. Knowledge representation in a workstation for reservior analysis. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 113-159.

[57] Lee, S. and Shin, K. G., 1987. Uncertain inference using belief functions. *Proceedings of the Third Conference on Artificial Intelligence Applications*, IEEE Computer Society, pp. 238-243.

[58] Lindley, D. V., 1987. The probability approach to the treatment of uncertainty in artificial intelligence and expert systems. Statistical Science, Vol. 2, No. 1, pp. 17-24.

[59] Lu, S. Y., and Stephanou, H. E., 1984. A set-theoretic framework for the processing of uncertain knowledge. AAAI-84 Proceedings, pp. 216-221.

[60] Maida, A. S., 1987. Frame theory. In Shapiro, S. C., Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[61] Marefat, M., Feghhi, S. J. and Kashyap, R. L., 1990. IDP: automating the CAD/CAM by reasoning about shape. In Proceedings, IEEE sixth conference on artificial intelligence applications.

[62] Maren, A. J., Harston, G. T., and Pap, R. M., 1990. Handbook of Neural Computing Applications. Academic Press, San Diego, USA.

[63] McCommon, R. B., 1988. PROSPECTOR III: towards a map-based expert system for regional mineral resource assessment. working paper, U. S. Geological Survey.

[64] McCammon, R. B., 1986. The muPROSPECTOR mineral consultant system. U.S. Geological Survey Bulletin 1697.

[65] McCornmack, M. D., 1991. Neural computing in geophysics. Geophysics: The leading edge of exploration, Vol. 10, No. 1, pp. 11-15.

[66] Miller, B. M., 1987. The muPETROL expert system for classifying world sedimentary basins. U.S. Geological Survey Bulletin 1810.

[67] Miller, B. M., 1990. An object-oriented expert system for sedimentary basin analysis with applications in petroleum geology. AAPG Bulletin, v. 74, pp. 719.

[68] Miller, R. G., 1968. Jackknife Variances. Ann. Math. Statist., vol. 39, pp. 567-582.

[69] Miller, R. G., 1974. The jackknife - a review. Biometrika, vol. 61, pp. 1-15.

[70] Moller-Jensen, L., 1990. Knowledge-based classification of an urban area using texture and context information in Landsat-TM imagery. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 899-904.

[71] Moon, W. M., 1990. Integration of geophysical and geological data using evidential belief function. IEEE Transactions, Geoscience and Remote Sensing, Vol. 28, pp. 711-720.

[72] Moon, W. M., Chung, C. F., and An, P., 1991. Representation and integration of geological, geophysical and remote sensing data. Geoinformatics. Vol. 2, No. 2, pp. 177-188.

[73] Ng, K. and Abramson, B., 1990. Uncertainty management in expert systems. IEEE Expert, Vol. 5, No. 2, pp. 29-48.

[74] Nielson, E., and Graham, D. C., 1985. Preliminary results of till petrographical and till geochemical studies at Farley Lake. Geological Survey, Manitoba Energy and Mines, Open File Report OF86-2.

[75] Pearl, J., 1985. Fusion, propagation, and structuring in Bayesian networks. tech. report No. CSD-850022, Computer Science Department, University of California,

Los Angeles. Pearl, J., 1987. AND/OR graph. In Shapiro, S. C., Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[76] Penas, C., Hadsell, F., and Stout, J., 1986 An expert system for determining vibroseis field parameters. Geobyte, winter, PP. 42-44.

[77] Pinsent, R. H., 1980. Nickle-copper mineralization in the Lynn Lake gabbro. Department of Energy and Mines, Provencial Government of Manitoba, Canada. Economic Geology Report, ER79-3.

[78] Quenouille, M. H., 1956. Notes on bias in estimation. Biometrika, vol. 43, pp. 353-360.

[79] Reboh, R., Reiter, J., and Gaschnig, J., 1982. Development of a knowledge-based interface to a hydrological simulation program. SRI Technical Report 3477, SRI International, Menlo Park, CA.

[80] Reddy, R. K. T. and Bonham-Carter, G. F., 1991. A decision-tree approach to mineral potential mapping in Snow Lake area, Manitoba. Canadian Journal of Remote Sensing, Vol. 17, No. 2, pp. 191-200.

[81] Rhee, J., 1991. A neural-net knowledge-based system with instance-based rules. Ph.D thesis abstract, University of California, Berkeley. In Dissertation Abstract International, Series B, Vol. 52, No. 4, pp. 2235.

[82] Rich, E. A., 1983. Aritificial intelligence. McGraw-Hill, New York.

[83] Rich, E. A., 1987. Artificial intelligence. In Shapiro, S. C., ed., Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[84] Schenk, T., and Zilberstein, O., 1990. Experiments with a rule-based system for interpreting linear map features. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 911-917.

[85] Shafer, G., 1987. Probability judgement in artificial intelligence and expert systems. Statistical Science, Vol. 2, No. 1, pp. 3-16.

[86] Shafer, G., and Logan, R., 1985. Implementing Dempster's rule for hierarchical evidence. working paper No. 174, School of Business, University of Kansas, U.S.A.

[87] Shafer, G., 1984. The problem of dependent evidence. working paper No. 164, School of Business, University of Kansas, U.S.A.

[88] Shafer, G., 1976. A mathematical theory of evidence. Princeton, N.J.: Princeton University Press.

[89] Shapiro, S. C., ed., 1987. Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[90] Shenoy, P. P., and Shafer, G., 1986. Propagating belief functions with local computation. IEEE Expert, Vol. 1, pp. 43-52.

[91] Shultz, A. W., Fang, J. H., Burston, Chen, H. C., and Reynold, S., 1988. XEOD: an expert system for determining clastic depositional environments. Geobyte, May, pp. 22-30.

[92] Slagle, J. and Gini, M., 1987. Problem reduction. In Shapiro, S. C., ed., Encyclopedia of artificial intelligence. Wiley-Interscience Publication.

[93] Singer, D. A., and Kouda, R., 1988. Integrating spatial and frequency information in search for Kuroko deposits of the Hokuroko district, Japan. Economic Geology, Vol. 83, No. 1, pp. 18-29.

[94] Spiegelhalter, D. J., 1987. Probabilistic expert systems in medicine: practical issues in handling uncertainty. Statistical Science, Vol. 2, No. 1, pp. 25-30.

[95] Syme, E. C., 1985. Geochemistry of metavolcanic rocks in the Lynn Lake Belt. Manitoba Mineral Resources Division, Geological Report GR84-1.

[96] Tukey, J. W., 1958. Bias and confidence in not-quite large samples (abstract). Ann. Math. Statist., vol. 29, pp. 614.

[97] Veezhinathan, J., Wanger, D. and Ehlers, J., 1991 First break picking using a neural network. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 179-202.

[98] Visher, G. S., and Sutterlin, P. G., 1990. De-Xpert: an expert systems approach to identifying depositional systems. Geobyte, October, pp. 57-62.

[99] Walker, M. G., 1988, Expert systems in geological exploration: can they be cost effective? Geobyte, August, pp. 18-20.

[100] Wang F., 1989. A fuzzy expert system for remote sensing image analysis. Proceedings of IGARSS'89, IEEE 89CH2768-0, pp. 848-851.

[101] Waterman D. A., 1986. A guide to expert systems. Addison-Wesley Publishing Company.

[102] Watson, S. R., 1987. Comment. Statistical Science, Vol. 2, No. 1, pp. 30-32.

[103] Winston P. H., 1984. Artificial Intelligence (second edition). Addison-Wesley Publishing Company.

[104] Zadeh, L. A., 1965. Fuzzy sets. IEEE Information and Control, vol. 8, pp. 338-353.

[105] Zadeh, L. A., 1978. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1, pp. 3-28.

[106] Zadeh, L. A., 1986. A simple view of the Dempster-Shafer theory of evidence and its implication for the rule combination. AI Magazine, Vol. 7, No. 2.

[107] Zhang, Z., and Simaan, M., 1991. Knowledge-based reasoning in SEISIS–a rule-based system for interpretation of seismic sections based on texture. In Aminzadeh, F. and Simaan, M., eds. Expert systems in exploration. Geophysical development series, Vol. 3. Society of Exploration Geophysicists, pp. 141-159.

[108] Zimmermann, H., and Zysno, P., 1980. Latent connectives, human decision making. Fuzzy Sets and Systems, vol. 4, pp. 37-51.

[109] Zimmermann H., 1985. Fuzzy set theory and its application. Kluwer Academic Publishers, pp. 37.

Appendix A: Objects, Data and Support Files

Objects Currently Available in the System

The present system has one top node, eight middle nodes, and fourteen terminal nodes. Table A.1 lists all the nodes available.

The Data and Support Files

The Data File Format

The data files are pixel matrix files in ASCII format. The pixel values can be numbers or letters. The dimensions of all the data files must be equal. The minimum dimension of the data matrix is 1x1 and the maximum depends on the machine and its configuration on which the program is running. An example of a 4x5 symbolic data file is given below.

```
3 2 3 7 5
3 3 7 5 2
3 3 7 5 5
1 1 7 5 5
```

The Legend Files

A legend file is used to describe the meanings of the values in a data file. Each symbolic data file must have a corresponding legend file. A numeric data file does not need a legend file. The format of the legend files is as below.

Type	Data	Description
Terminal Objects	air_magnet	airborne magnetic total field
	step_airmag	airborne magnetic intensity anomaly
	ground_mag.	ground magnetic intensity anomaly
	geology	geology map
	Input	airborne INPUT map
	em_grade	airborne electromagnetic anomaly
	lake	surface water map
	resistivity	ground resistivity
	vlf1	VLF using source 1
	vlf2	VLF using source 2
	chargeability	IP chargeability
	image1	band 1 MEIS image
	image2	band 2 MEIS image
	ground_em	ground EM anomaly
Middle Objects	high_airmag	favorable air magnetic anomaly
	susceptibility	favorable susceptibility
	air_em	airborne EM anomaly
	vlf	VLF anomaly
	conductor	geological body of high conductivity
	high_con	favorable conductor
	fav_geol	Favorable Geological condition
	image	favorable visual evidence
Top Object	target	a base metal deposit

Table 7.1: Available Objects in the System

meaning: value

where *value* is a pixcel value in the data file, which can be a number or character; *meaning* is a character string which describes the meaning of the *value* and it is known by the corresponding rule-base. An example is given below.

strong_anomaly: 14
medium_anomaly: 11
weak_anomaly: 8
no_anomaly: 4
nodata: 15
end:

Note that there is no space at beginning of each line and the file is ended by an "end:".

The Project File

The project file tells the system the addresses of data files, corresponding legend information and rule bases. The format for a terminal object is as below;

object_name,data_file,legend_file,rule_base,

where *object_name* is the name of an object in table 7; *data_file* is the path to find corresponding data file; *legend_file* is the path to find legend information file of the data file; *rule_base* is the path of the rule base of the object. they are all character strings with less than 30 characters.

The format for a middle node is given below;

object_name,rule_base,

where *object_name* and *rule_base* have the same meaning as described above.

The format of the top node is

object_name,

where *object_name* is the name of the top object in table 7.

An example is given below;

geology,./sdata/sgeol,./slegend/geol.inf,./rule/geol.rul,

air_magnet,./sdata/sairmag,./rule/airmag.rul,

lake,./sdata/slake,./slegend/lake.inf,./rule/lake.rul,

em_grade,./sdata/sairem,./slegend/airem.inf,./rule/emgd.rul,

ground_em,./sdata/sgem,./slegend/agem.inf,./rule/gndem.rul,

ground_mag,./sdata/sgmag,./slegend/agmag.inf,./rule/gndmag.rul,

step_airmag,./sdata/sstepmag,./slegend/astepmag.inf,./rule/step_airmag.rul,

high_airmag,./rule/high_airmag.rul,

conductor,./rule/cndt.rul,

high_con,./rule/hcon.rul,

air_em,./rule/airem.rul,

fav_geol,./rule/favg.rul,

susceptibility,./rule/suscep.rul,

target,

end,

note that the file is ended with an "end,".

Customizing the System Knowledge

The knowledge of the system can be customized or updated by modifying the relation network file and rule base files.

The Relation Network File

The relation network file defines the evidential relation among the objects. The format of connection definition of a node is given as below.

object_name: n

object1,level,relation,object2,

...

objectn,level,relation,object1,

The *object_name* is the name of an object that where connections are being defined. *n* is the number of connections to the object. It is followed by *n* lines each of which defines a connection. *object1* is the name of the first object connected to *object_name*. *level* is the level of *object1* relative to *object_name*. It can be "up" if *object1* is a higher level object of *object_name* or "down" if it is a lower level object of *object_name*. *relation* defines the relation between *object1* and *object2*. It can be "and" or "or". If there is only one lower level object, it should be "single". *object2* is the name of the second object connected to *object_name*. It is "nil" if there is only one lower level object.

The Rule Base Files

The rules in a rule file are in the format

condition,

conclusion, bel dis unc

where *condition* and *conclusion* are character strings; *bel* is the lower probability or degree of belief to which the rule is true; *dis* is the probability to its negation or degree of belief to which the rule is false; and *unc* is the degree of uncertainty.

Appendix B

Source Code of the Expert System in C^{++}

```
//This is the source code of the prototype system in C++.
//It is developed using AT&T C++, version 2.01.
//For compilation, put all included files in the specified
//directories and simply use CC gin.

// All other files are divided using //****************//

// File gin --- manage all network objects.

#include "etc/util.h"
#include "etc/network.h"
#include "etc/globe.h"
#include "header/target.h"
#include "header/conductor.h"
#include "header/fav_geol.h"
#include "header/air_em.h"
#include "header/groundem.h"
#include "header/em_grade.h"
#include "header/input.h"
#include "header/geol.h"
#include "header/ground_mag.h"
#include "header/air_magnet.h"
#include "header/charge.h"
#include "header/resis.h"
#include "header/vlf.h"
#include "header/vlf1.h"
#include "header/vlf2.h"
#include "header/suscep.h"
#include "header/lake.h"
#include "header/high_con.h"
#include "header/high_airmag.h"
#include "header/step_airmag.h"
#include "header/img.h"
#include "header/img1.h"
#include "header/img2.h"

static error_handler merr("meta error: ");

class meta{
    int r, c;
//    bfmatrix *B;
    geology *geol;
    Input *inpt;
    lake *lke;
    high_con *hcon;
    em_grade *emgrd;
    ground_em *grndem;
    conductor *cndctr;
```

```
    fav_geol *favgeol;
    air_em *airem;
    resistivity *resis;
    image *img;
    image1 *img1;
    image2 *img2;
    vlf1 *vlfa;
    vlf2 *vlfs;
    vlf *vf;
    susceptibility *suscep;
    chargeability *charge;
    air_magnet *airmag;
    ground_mag *grndmag;
    target *tgt;
    step_airmag *sairmag;
    high_airmag *hairmag;
    char *get_string(FILE *);
    susceptibility *set_suscep(char *, char *, network &,
                    int, int);
    resistivity *set_resis(char *, char *, char *, char *,
                network &, int, int);
    chargeability *set_charge(char *, char *, char *, char *,
                network &, int, int);
    step_airmag *set_sairmag(char *, char *, char *, char *,
                network &, int, int);
    vlf1 *set_vlf1(char *, char *, char *, char *,
                network &, int, int);
    vlf2 *set_vlf2(char *, char *, char *, char *,
                network &, int, int);
    geology *set_geol(char*, char*, char*, char*, network &,
                int, int);
    air_em *set_airem(char *, char *, network &, int, int);
    vlf *set_vlf(char *, char *, network &, int, int);
    image *set_img(char *, char *, network &, int, int);
    ground_mag *set_grndmag(char *, char *, char *, char *,
                network &, int, int);
    air_magnet *set_airmag(char *, char *, char *,
                network &, int, int);
    image1 *set_img1(char *, char *, char *,
                network &, int, int);
    image2 *set_img2(char *, char *, char *,
                network &, int, int);
    Input *set_input(char*, char*, char*, char*,
           network &, int, int);
    lake *set_lake(char*, char*, char*, char*,
           network &, int, int);
    em_grade *set_emgrade(char*, char*, char*, char*,
           network &, int, int);
    ground_em *set_grndem(char*, char*, char*, char*,
           network &, int, int);
    conductor *set_cndctr(char*, char*, network &, int, int);
    high_con *set_hcon(char*, char*, network &, int, int);
    high_airmag *set_hairmag(char*, char*, network &, int, int);
    fav_geol *set_favgeol(char*, char*, network &, int, int);
    target *set_tgt(char*, network &, int, int);
public:
    void print_result();
    meta(network &, char *, int, int);
    bfmatrix *infer(network &, char *, char *);
    void inference(network &, char *);
    void passto(char *, bfmatrix *);
```

```
    bfmatrix *apply_rules(char *, char *);
    void fpri(FILE *, FILE *);
    void del_obj(char *);
};

meta::meta(network &nw, char *metaf, int row, int col){
      r = row;
      c = col;
      int x = row;
      int y = col;
      FILE *fptr;
      char *nam, *datf, *rulf, *inf;
      int flag = 0;
      if((fptr = fopen(metaf, "r")) == NULL)
          merr.terminate("can not open %s file for file locations\n", metaf);
      do {
        nam = get_string(fptr);
        printf("name nam = %s\n", nam);
      if((strcmp(nam, "end")) != 0){
        if
          ((strcmp(nam, "geology")) == 0){
            datf = get_string(fptr);
            inf = get_string(fptr);
            rulf = get_string(fptr);
     printf("The data size is %d * %d", x, y);
            geol = set_geol(nam, datf, inf, rulf, nw, x, y);
}
        else if
          ((strcmp(nam, "Input")) == 0){
           datf = get_string(fptr);
           inf = get_string(fptr);
           rulf = get_string(fptr);
           inpt = set_input(nam, datf, inf, rulf, nw, x, y);
         }
        else if
          ((strcmp(nam, "lake")) == 0){
           datf = get_string(fptr);
           inf = get_string(fptr);
           rulf = get_string(fptr);
           lke = set_lake(nam, datf, inf, rulf, nw, x, y);
         }
        else if
          ((strcmp(nam, "em_grade")) == 0){
           datf = get_string(fptr);
           inf = get_string(fptr);
           rulf = get_string(fptr);
           emgrd = set_emgrade(nam, datf, inf, rulf, nw, x, y);
         }
        else if
          ((strcmp(nam, "ground_em")) == 0){
           datf = get_string(fptr);
           inf = get_string(fptr);
           rulf = get_string(fptr);
           grndem = set_grndem(nam, datf, inf, rulf, nw, x, y);
         }
        else if
          ((strcmp(nam, "air_magnet")) == 0){
           datf = get_string(fptr);
```

```
//            inf = get_string(fptr);
             rulf = get_string(fptr);
      printf("The data size is %d * %d", x, y);
             airmag = set_airmag(nam, datf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "image1")) == 0){
             datf = get_string(fptr);
//             inf = get_string(fptr);
             rulf = get_string(fptr);
             img1 = set_img1(nam, datf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "image2")) == 0){
             datf = get_string(fptr);
//             inf = get_string(fptr);
             rulf = get_string(fptr);
             img2 = set_img2(nam, datf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "ground_mag")) == 0){
             datf = get_string(fptr);
             inf = get_string(fptr);
             rulf = get_string(fptr);
             grndmag = set_grndmag(nam, datf, inf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "vlf1")) == 0){
             datf = get_string(fptr);
             inf = get_string(fptr);
             rulf = get_string(fptr);
             vlfa = set_vlf1(nam, datf, inf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "vlf2")) == 0){
             datf = get_string(fptr);
             inf = get_string(fptr);
             rulf = get_string(fptr);
             vlfs = set_vlf2(nam, datf, inf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "resistivity")) == 0){
             datf = get_string(fptr);
             inf = get_string(fptr);
             rulf = get_string(fptr);
             resis = set_resis(nam, datf, inf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "chargeability")) == 0){
             datf = get_string(fptr);
             inf = get_string(fptr);
             rulf = get_string(fptr);
             charge = set_charge(nam, datf, inf, rulf, nw, x, y);
           }
          else if
            ((strcmp(nam, "step_airmag")) == 0){
             datf = get_string(fptr);
```

```
            inf = get_string(fptr);
            rulf = get_string(fptr);
            sairmag = set_sairmag(nam, datf, inf, rulf, nw, x, y);
          }
         else if
           ((strcmp(nam, "conductor")) == 0){
            rulf = get_string(fptr);
            cndctr = set_cndctr(nam, rulf, nw, x, y);
          }
         else if
           ((strcmp(nam, "high_con")) == 0){
            rulf = get_string(fptr);
            hcon = set_hcon(nam, rulf, nw, x, y);
          }
         else if
           ((strcmp(nam, "high_airmag")) == 0){
            rulf = get_string(fptr);
            hairmag = set_hairmag(nam, rulf, nw, x, y);
          }
         else if
           ((strcmp(nam, "fav_geol")) == 0){
            rulf = get_string(fptr);
            favgeol = set_favgeol(nam, rulf, nw, x, y);
          }
         else if
           ((strcmp(nam, "target")) == 0){
            tgt = set_tgt(nam, nw, x, y);
          }
         else if
           ((strcmp(nam, "air_em")) == 0){
              rulf = get_string(fptr);
              airem = set_airem(nam, rulf, nw, x, y);
    }
         else if
           ((strcmp(nam, "vlf")) == 0){
              rulf = get_string(fptr);
              vf = set_vlf(nam, rulf, nw, x, y);
    }
         else if
           ((strcmp(nam, "image")) == 0){
              rulf = get_string(fptr);
              img = set_img(nam, rulf, nw, x, y);
    }
         else if
           ((strcmp(nam, "susceptibility")) == 0){
              rulf = get_string(fptr);
              suscep = set_suscep(nam, rulf, nw, x, y);
    }
         else
           merr.terminate("sorry, I have no idea about %s",nam);
       while((c = fgetc(fptr)) != '\n'){;}
      }
     else
       flag = 1;
      }while(flag == 0);
   fclose(fptr);
//   write_dat();
 }
```

```
//void meta::print_result(){
//    B->print();
//  }

geology *meta::set_geol(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
   printf("x and y are %d and %d \n", x, y);
   geology *tp = new geology(nam, datf, inf, rulf, nw, x, y);
   return tp;
 }

Input *meta::set_input(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  Input *tp = new Input(nam, datf, inf, rulf, nw, x, y);
   return tp;
}

lake *meta::set_lake(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  lake *tp = new lake(nam, datf, inf, rulf, nw, x, y);
   return tp;
}

em_grade *meta::set_emgrade(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  em_grade *tp = new em_grade(nam, datf, inf, rulf, nw, x, y);
   return tp;
}

ground_em *meta::set_grndem(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  ground_em *tp = new ground_em(nam, datf, inf, rulf, nw, x, y);
   return tp;
}

resistivity *meta::set_resis(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  resistivity *tp = new resistivity(nam, datf, inf, rulf, nw, x, y);
   return tp;
}

chargeability *meta::set_charge(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  chargeability *tp = new chargeability(nam, datf,inf,rulf,nw,x,y);
   return tp;
}

step_airmag *meta::set_sairmag(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  step_airmag *tp = new step_airmag(nam, datf,inf,rulf,nw,x,y);
   return tp;
}

vlf1 *meta::set_vlf1(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  vlf1 *tp = new vlf1(nam, datf,inf,rulf,nw,x,y);
   return tp;
}
```

```
vlf2 *meta::set_vlf2(char *nam, char *datf, char *inf,
        char *rulf, network &nw, int x, int y){
  vlf2 *tp = new vlf2(nam, datf,inf,rulf,nw,x,y);
   return tp;
}


conductor *meta::set_cndctr(char *nam, char *rulf,
            network &nw, int x, int y){
  conductor *tp = new conductor(nam, rulf, nw, x, y);
   return tp;
}

high_con *meta::set_hcon(char *nam, char *rulf,
            network &nw, int x, int y){
  high_con *tp = new high_con(nam, rulf, nw, x, y);
   return tp;
}

high_airmag *meta::set_hairmag(char *nam, char *rulf,
            network &nw, int x, int y){
  high_airmag *tp = new high_airmag(nam, rulf, nw, x, y);
   return tp;
}

vlf *meta::set_vlf(char *nam, char *rulf,
            network &nw, int x, int y){
  vlf *tp = new vlf(nam, rulf, nw, x, y);
   return tp;
}
image *meta::set_img(char *nam, char *rulf,
            network &nw, int x, int y){
  image *tp = new image(nam, rulf, nw, x, y);
   return tp;
}

air_em *meta::set_airem(char *nam, char *rulf,
            network &nw, int x, int y){
  air_em *tp = new air_em(nam, rulf, nw, x, y);
   return tp;
}

air_magnet *meta::set_airmag(char *nam, char *datf,
            char *rulf, network &nw, int x, int y){
  air_magnet *tp = new air_magnet(nam, datf, rulf, nw, x, y);
   return tp;
}
image1 *meta::set_img1(char *nam, char *datf,
            char *rulf, network &nw, int x, int y){
  image1 *tp = new image1(nam, datf,rulf,nw,x,y);
   return tp;
}
image2 *meta::set_img2(char *nam, char *datf,
            char *rulf, network &nw, int x, int y){
  image2 *tp = new image2(nam, datf,rulf,nw,x,y);
   return tp;
}
```

```
ground_mag *meta::set_grndmag(char *nam, char *datf, char *inf,
            char *rulf, network &nw, int x, int y){
  ground_mag *tp = new ground_mag(nam, datf,inf,rulf,nw,x,y);
   return tp;
}

fav_geol *meta::set_favgeol(char *nam, char *rulf,
                network &nw, int x, int y){
  fav_geol *tp = new fav_geol(nam, rulf, nw, x, y);
   return tp;
}

susceptibility *meta::set_suscep(char *nam, char *rulf,
                network &nw, int x, int y){
  susceptibility *tp = new susceptibility(nam, rulf, nw, x, y);
   return tp;
}

target *meta::set_tgt(char *nam, network &nw, int x, int y){
  target *tp = new target(nam, nw, x, y);
   return tp;
}

//void meta::write_dat(){
//     geol->write_map("geology data");
//     inpt->write_map("input data");
//   }

char *meta::get_string(FILE *pt){
     char ch[30];
     int c;
     int i;
     char *s;
        for(i=0; i<30; i++)
          ch[i] = '\0';
        i = 0;
        while((c = fgetc(pt)) != ','){
          ch[i] = c;
          i++;
        if(i > 29)
         merr.terminate("node or file name is longer than 30 chars");
       }
        s = malloc(strlen(ch));
        strcpy(s, ch);
        return s;
   }

void meta::inference(network &nw, char *ndnm){
      char *tw;
      printf("I am considering %s\n", ndnm);
      nw.write_netwk();
      tw = nw.pick_towho(ndnm);
      printf("%s's lower node is %s\n", ndnm, tw);
     if((strcmp(tw, "none")) == 0)
        merr.terminate("%s has no lower node\n", ndnm);
     else{
//       bfmatrix *tmp = new bfmatrix(r,c);
      printf("I am considering node %s\n", ndnm);
        infer(nw, ndnm, tw);
//       B = tmp;
```

```
    }
  };

void meta::fpri(FILE *bpt, FILE *dpt){
     for (int i=0; i<r; i++)
       for (int j=0; j<c; j++){
         if (tgt->B->bel->val(i,j) < 0.0005)
              tgt->B->bel->val(i,j) = 0.0;
         if (tgt->B->dis->val(i,j) < 0.0005)
              tgt->B->dis->val(i,j) = 0.0;
      };
     tgt->B->bel->fpri(bpt);
     tgt->B->dis->fpri(dpt);
  }

bfmatrix *meta::infer(network &nw, char *ndnm, char *tw){
                  //ndnm must not be a terminal node
                  //tw is towho of ndnm
    int y;
    bfmatrix *temp = new bfmatrix(r, c);
//    nw.write_node(nw.getnd(ndnm));
    char *ww, *h, *nm, *tow, *nde;
    printf("before tmnl_or_ndwn %s\n",tw);
    y = nw.tmnl_or_ndwn(tw);
    printf("y = %d\n",y);
    if((nw.tmnl_or_ndwn(tw)) == 1){
      printf("this is t_orn node\n");
      temp = apply_rules(ndnm, tw);
//      temp->print();
      nw.set_visited(ndnm, tw);
      del_obj(tw);
//      nw.write_node(nw.getnd(ndnm));
      }
    else{ //recurse to lower node
      nm = malloc(strlen(tw)+1);
      strcpy(nm, tw);
      printf("In else\n");
      tow = nw.pick_towho(nm);
      printf("Now considering %s and %s\n", nm,tow);
//      nw.write_node(nw.getnd(ndnm));
//      nw.write_node(nw.getnd(nm));
      infer(nw, nm, tow);
//      nw.set_visited(nm, tow);
//      del_obj(tow);
      temp = apply_rules(ndnm, tw);
      nw.set_visited(ndnm, tw);
      del_obj(tw);
     }
//search other conn at same level.
    ww = nw.get_withwho(ndnm, tw);
    nde = malloc(strlen(tw)+1);
    strcpy(nde, tw);
    while((strcmp(ww, "nil")) != 0 && //there are other conn in
          nw.visitedp(ndnm, ww) == 0){ //and not visited
       if ((nw.yesp(ndnm, ww)) == 1){ //if yes, infer
          h = nw.get_how(ndnm, nde);
         printf("how = %s for %s with %s\n",h,nde,ww);
         if((strcmp(h, "and")) == 0)
```

```
            temp = and(temp, infer(nw, ndnm, ww));
          else if
            ((strcmp(h, "or")) == 0)
             temp = or(temp, infer(nw, ndnm, ww));
          else
              merr.terminate("how is not and or or");
  }
//          nw.set_visited(ndnm, ww);
          nde = realloc(nde, strlen(ww)+1);
          strcpy(nde, ww);
          ww = nw.get_withwho(ndnm, ww);
        }
      passto(ndnm, temp);
      return temp;
}

void meta::del_obj(char *cnm){
   printf("object %s deleted\n", cnm);
   if((strcmp(cnm, "geology")) == 0){
        delete geol;
     }
   else if
     ((strcmp(cnm, "em_grade")) == 0){
        delete emgrd;
     }
   else if
     ((strcmp(cnm, "ground_em")) == 0){
        delete grndem;
     }
   else if
     ((strcmp(cnm, "Input")) == 0){
        delete inpt;
     }
   else if
     ((strcmp(cnm, "lake")) == 0){
        delete lke;
     }
   else if
     ((strcmp(cnm, "high_con")) == 0){
        delete hcon;
     }
   else if
     ((strcmp(cnm, "high_airmag")) == 0){
        delete hairmag;
     }
   else if
     ((strcmp(cnm, "conductor")) == 0){
        delete cndctr;
     }
   else if
     ((strcmp(cnm, "fav_geol")) == 0){
        delete favgeol;
     }
   else if
     ((strcmp(cnm, "air_em")) == 0)
        delete airem;
   else if
     ((strcmp(cnm, "air_magnet")) == 0)
        delete airmag;
   else if
     ((strcmp(cnm, "image1")) == 0)
        delete img1;
```

```
    else if
      ((strcmp(cnm, "image2")) == 0)
          delete img2;
    else if
      ((strcmp(cnm, "vlf2")) == 0)
          delete vlfs;
    else if
      ((strcmp(cnm, "image")) == 0)
          delete img;
    else if
      ((strcmp(cnm, "vlf1")) == 0)
          delete vlfa;
    else if
      ((strcmp(cnm, "ground_mag")) == 0)
          delete grndmag;
    else if
      ((strcmp(cnm, "susceptibility")) == 0)
          delete suscep;
    else if
      ((strcmp(cnm, "resistivity")) == 0)
          delete resis;
    else if
      ((strcmp(cnm, "chargeability")) == 0)
          delete charge;
    else if
      ((strcmp(cnm, "step_airmag")) == 0)
          delete sairmag;
    else if
      ((strcmp(cnm, "vlf")) == 0)
         delete vf;
    else
      merr.terminate("Can't delete node %s\n", cnm);
 }


bfmatrix *meta::apply_rules(char *ndnm, char *cnm){
   printf("apply_rules for %s, %s called\n", ndnm,cnm);
//   bfmatrix *temp = new bfmatrix(r, c);
   if((strcmp(cnm, "geology")) == 0){
       return (geol->apply_rules(ndnm));
     }
   else if
     ((strcmp(cnm, "em_grade")) == 0){
//        printf("emgrd->apply_rules(%s) called\n",ndnm);
       return (emgrd->apply_rules(ndnm));
     }
   else if
     ((strcmp(cnm, "ground_em")) == 0){
       return (grndem->apply_rules(ndnm));
     }
   else if
     ((strcmp(cnm, "Input")) == 0){
       return (inpt->apply_rules(ndnm));
     }
   else if
     ((strcmp(cnm, "conductor")) == 0){
       return (cndctr->apply_rules(ndnm));
     }
   else if
     ((strcmp(cnm, "high_con")) == 0){
       return (hcon->apply_rules(ndnm));
```

```
    }
  else if
    ((strcmp(cnm, "high_airmag")) == 0){
      return (hairmag->apply_rules(ndnm));
    }
  else if
    ((strcmp(cnm, "lake")) == 0){
      return (lke->apply_rules(ndnm));
    }
  else if
    ((strcmp(cnm, "fav_geol")) == 0){
      return (favgeol->apply_rules(ndnm));
    }
  else if
    ((strcmp(cnm, "air_em")) == 0)
      return (airem->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "image1")) == 0)
      return (img1->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "image2")) == 0)
      return (img2->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "air_magnet")) == 0)
      return (airmag->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "vlf2")) == 0)
      return (vlfs->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "vlf1")) == 0)
      return (vlfa->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "ground_mag")) == 0)
      return (grndmag->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "susceptibility")) == 0)
      return (suscep->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "resistivity")) == 0)
      return (resis->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "chargeability")) == 0)
      return (charge->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "step_airmag")) == 0)
      return (sairmag->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "vlf")) == 0)
      return (vf->apply_rules(ndnm));
  else if
    ((strcmp(cnm, "image")) == 0)
      return (img->apply_rules(ndnm));
  else
    merr.terminate("node %s unkmown.\n", cnm);
//  return temp;
 }

void meta::passto(char *ndnm, bfmatrix *M){
 //  (*M).print();
   if((strcmp(ndnm, "fav_geol")) == 0){
```

```
        favgeol->B = M;
      }
    else if
      ((strcmp(ndnm, "conductor")) == 0){
        cndctr->B = M;
      }
    else if
      ((strcmp(ndnm, "high_airmag")) == 0){
        hairmag->B = M;
      }
    else if
      ((strcmp(ndnm, "high_con")) == 0){
        hcon->B = M;
      }
    else if
      ((strcmp(ndnm, "target")) == 0){
        tgt->B = M;
      }
    else if
      ((strcmp(ndnm, "air_em")) == 0){
        airem->B = M;
      }
    else if
      ((strcmp(ndnm, "vlf")) == 0){
        vf->B = M;
      }
    else if
      ((strcmp(ndnm, "image")) == 0){
        img->B = M;
      }
    else if
      ((strcmp(ndnm, "susceptibility")) == 0){
        suscep->B = M;
      }
    else
       merr.terminate("can't pass bfmatrix to %s\n", ndnm);
    printf("bfmatrix passed to %s\n", ndnm);
//    delete M;
 }

main(){
    int row, col, i, c;
    char ch[100];

    printf("Welcome to this prototype expert system.\n");
    printf("Please enter your project file name: ");
    i=0;
    while((c = getchar()) != '\n'){
      ch[i]=c;
      i++;
     }
    printf("Your project file is %s.\n", ch);
    printf("Please enter the number of rows of your data: ");
    scanf("%d",&row);
    printf("Please enter the number of columns of your data: ");
    scanf("%d", &col);
    printf("Your data file size is %d * %d\n", row, col);

    network nw("./rule/network", "target");
    meta mt(nw, ch, row, col);
    printf("all are set up\n");
    mt.inference(nw, "target");
```

```
    FILE *bpt = fopen("sbel", "w");
    FILE *dpt = fopen("sdis", "w");
    mt.fpri(bpt, dpt);
  };

//*************************************************************//

// util.h  ---- a set of utility programs.

#include <strings.h>
#include <stdio.h>
#include <stdarg.h>
#include <stream.h>
#include <stdlib.h> //exit()

//error.h a class to perform error management inside
//other classes

class error_handler{
  char *msg;
public:
  error_handler(char *class_msg){
        msg = class_msg;
  }
  void message(char *format_string,...);
  void terminate(char *format_string,...);
};
//error.c implemented using ANSI C library functions


void error_handler::message(char *format_string,...){
  va_list arg_pointer;
  va_start(arg_pointer, format_string);
  fprintf(stderr, "%s ", msg); //print class message first
  vfprintf(stderr, format_string, arg_pointer);
  va_end(arg_pointer);
}

void error_handler::terminate(char *format_string,...){
  va_list arg_pointer;
  va_start(arg_pointer, format_string);
  fprintf(stderr, "%s ", msg); //print class message first
  vfprintf(stderr,format_string, arg_pointer);
  va_end(arg_pointer);
  exit(1);
}

//matrix.h

static error_handler err("matrix error: ");

//a structure to hold matrix information.

struct matrep{
    double **m; //pointor yto the matrix
    int r,c; //number of rows and colums
    int n;  //reference count
};

class matrix{
    matrep *p;
```

```
    void error(char *msg1, char *msg2=""); //private fun.
  public:
    matrix(int rows=1, int columns=1, double initval=0);
    matrix(matrix& x); //copy initializer
    ~matrix();
    void readat(char *datfile);//read data from afile
    matrix operator=(const matrix& rval);//matrix assign
    matrix operator+(const matrix& rval);//matrix add
    matrix operator*(const matrix& rval);//matrix multi
    matrix operator/(const matrix& rval);//matrix divide
    matrix operator-(const matrix& rval);//matrix minus
    double &val(int row, int col){//element selection
           //can be used to read and write an element
//    if (row< p->r && col<p->c)
      return (p->m[row][col]);
//    else
//       err.terminate("val index out of range\n");
}
    void change(int row, int col, double newval);
    double get(int row, int col);
    void print(char *msg="");
    void fpri(FILE *);
};
//matrix.c
void matrix::error(char *msg1, char *msg2){
    fprintf(stderr,"matrix error: %s %s\n",msg1,msg2);
    exit(1);
}

matrix::matrix(int rows, int columns, double initval){
     //create the structure
     p=new matrep;
       //allocate memory for the actual matrix;
     p->m = new double *[rows];
     for (int x=0; x< rows; x++)
        p->m[x] = new double[columns];
     p->n = 1;//so far there is one reference to this
            //data
     p->r = rows;
     p->c = columns;
     for (int i=0; i <rows; i++)
        for (int j=0; j< columns; j++)
            p->m[i][j]= initval;
}

matrix::matrix(matrix& x){
     x.p->n++;// adding another reference
     p= x.p; //point to the new matrep
}

double matrix::get(int rw, int cl){
     if(rw<p->r && cl<p->c)
        return (p->m[rw][cl]);
     else
       err.terminate("get out of range\n");
   }

matrix matrix::operator=(const matrix& rval){
    //clean up current value
```

```
     if(--p->n ==0) { //if nobody else is referencing us.
       for (int x=0; x< p->r; x++)
           delete p->m[x];
         delete p->m;
         delete p;
    }
    //connect to new value
    rval.p->n++;//tell the rval it has another refer.
    p=rval.p;// point at the rval matrep
    return *this;
}
matrix::~matrix(){
    if (--p->n == 0){ //if reference count goes to 0
      for (int x=0; x<p->r; x++)
          delete p->m[x];
        delete p->m; // delete data
      delete p;
     }
}

/*double & matrix::val(int row, int col){
     if (row< p->r && col<p->c)
        return (p->m[row][col]);
     else
        err.terminate("val index out of range\n");
}*/

void matrix::change(int row, int col, double newval){
     if (row< p->r && col<p->c)
        p->m[row][col] = newval;
     else
        err.terminate("index out of range\n");
}


void matrix::readat(char *datfile){
      printf("reading %d*%d data from %s\n", p->r, p->c, datfile);
      float s;
      FILE *in;
      if((in =fopen(datfile, "r")) == NULL)
        err.terminate("can't open %s for inupt\n", datfile);
      for(int i=0; i<p->r; i++)
        for(int j=0; j<p->c; j++){
          fscanf(in, "%g ", &s);
          p->m[i][j] = s;
        }
 //   print("datamatrix \n");
    fclose(in);
    }

matrix matrix::operator*(const matrix& rval){
     if(p->r != rval.p->r || p->c != rval.p->c)
       err.terminate("matrix dimesions are not equal");
      matrix result(p->r,rval.p->c);
      for(int row=0; row< p->r; row++)
         for(int col=0; col< rval.p->c; col++)
           result.p->m[row][col]
                =p->m[row][col]*rval.p->m[row][col];
      return result;
```

```
}

matrix matrix::operator/(const matrix& rval){
    if(p->r != rval.p->r || p->c != rval.p->c)
      err.terminate("matrix dimesions are not equal");
     matrix result(p->r,rval.p->c);
     for(int row=0; row< p->r; row++)
        for(int col=0; col< rval.p->c; col++)
          result.p->m[row][col]
             =p->m[row][col]/rval.p->m[row][col];
      return result;
  }

matrix matrix::operator-(const matrix& rval){
    if(p->r != rval.p->r || p->c != rval.p->c)
      err.terminate("matrix dimesions are not equal");
     matrix result(p->r,rval.p->c);
     for(int row=0; row< p->r; row++)
        for(int col=0; col< rval.p->c; col++)
          result.p->m[row][col]
             =p->m[row][col]-rval.p->m[row][col];
      return result;
  }

matrix matrix::operator+(const matrix& rval){
    if(p->r != rval.p->r || p->c != rval.p->c)
      err.terminate("matrix dimesions are not equal");
     matrix result(p->r,rval.p->c);
     for(int row=0; row< p->r; row++)
        for(int col=0; col< rval.p->c; col++)
          result.p->m[row][col]
               =p->m[row][col]+rval.p->m[row][col];
     return result;
}

void matrix::print(char *msg){
    int f = 1;
    printf("\n");
    if(*msg) printf("%s\n",msg);
    for(int row=0; row<p->r; row++){
      for(int col=0; col<p->c; col++){
        if (col == f * 15){
        printf("\n");
        f++;
      }
        printf("%4.4g ", p->m[row][col]);
      }
     f = 1;
     printf("\n");
  }
}

void matrix::fpri(FILE *fpt){
    int f = 1;
    for(int row=0; row<p->r; row++){
      for(int col=0; col<p->c; col++){
        if (col == f * 10){
        fprintf(fpt, "\n");
        f++;
```

```
      }
        fprintf(fpt, "%7.2g", p->m[row][col]);
      }
     f = 1;
     fprintf(fpt, "\n");
  }
}

class bfmatrix{
    int r, c;
  public:
    matrix *bel;
    matrix *dis;
//    matrix unc;
    bfmatrix(int rows=1, int cols=1, double initval=0);
    ~bfmatrix();
    void bfval(int row, int col, double b, double d);
    void print(){bel->print("bel"); dis->print("dis");
//            unc.print("unc");
     }
    int gr(){return r;}
    int gc(){return c;}
    void fpri(FILE *, FILE *);
 };

bfmatrix::bfmatrix(int rows, int cols, double initval){
        r = rows;
        c = cols;
        bel = new matrix(rows, cols, initval);
        dis = new matrix(rows, cols, initval);
//        matrix *U = new matrix(rows, cols, initval);
//        bel = *B;
//        dis = *D;
//        unc = *U;
 };

void bfmatrix::bfval(int row, int col, double b, double d){
//      printf("bfval called\n");
      if(row<r && col<c){
        bel->val(row, col) = b;
        dis->val(row, col) = d;
//        unc.val(row, col) = u;
//        printf("bel.val(%d,%d)=%g, dis.val(%d,%d)=%g\n",row,col,
//         bel.val(row,col),row,col,dis.val(row,col));
      }
      else
        err.terminate("bf index out of range\n");
 };

 bfmatrix::~bfmatrix(){
   delete bel;
   delete dis;
 };

void bfmatrix::fpri(FILE *bpt, FILE *dpt){
   bel->fpri(bpt);
   dis->fpri(dpt);
 };

// dynarray.h

#ifndef dynarray_h
```

```
#define dynarray_h

const chunk = 5;

class dynarray{
    void ** array;
    int size;
    int cursor;
    void error(char * msg = "");
  public:
    dynarray();
    ~dynarray();
    int add(void *);
    int remove(void *);
    int remove(int);
    void reset();
    void * next();
    int index(){return cursor;}
    void *current(){return array[cursor];}
    void *operator[](int);
    void *take(int);
    int count();
  };

#endif
//dameth.c  methods of dynarray

void dynarray::error(char * msg){
   printf("dynarray error: %s\n", msg);
   exit(1);
 }

   dynarray::dynarray(){
      array = new void*[chunk];
      for (int i=0; i<chunk; i++)
         array[i]=(void *)0;
      size=chunk;
      cursor=0;
    }

   dynarray::~dynarray(){
       delete array;
     }

void *dynarray::take(int x){
        if (x<0 || x>=size)
            error("take index out of order");
        return array[x];
      }

int dynarray::add(void * new_element){
      for(int i=0; i<size; i++)
         if(array[i] == (void*)0){
            array[i]=new_element;
            return i;
          }
    int tempsize = size+chunk;
    void ** temp = new void*[tempsize];
    for(i=0;i<size;i++)
      temp[i]=array[i];
    for(;i<tempsize;i++)
```

```
      temp[i]=(void*)0;
    temp[i=size]=new_element;
    delete array;
    array = temp;
    size = tempsize;
    return i;
}

int dynarray::remove(void* rp){
    for(int i=0; i<size; i++)
       if(array[i]==rp){
          array[i]=(void *)0;
          return 1;
        }
     return 0;
}

int dynarray::remove(int ri){
    if(ri<0 || ri>size)
      error("remove index out of range");
    if(array[ri]){
       array[ri]=(void*)0;
       return 1;
     }
     return 0;
}

void dynarray::reset(){
     cursor = 0;
     while((array[cursor] == (void *)0) && (cursor<size-1))
        cursor++;
   }

void * dynarray::next(){
     if(cursor == size-1)
        return (void *)0;
     while(array[++cursor] == (void *)0)
        if(cursor == size-1)
        return (void *)0;
     return array[cursor];
   }

void * dynarray::operator[](int x){
        if (x<0 || x>=size)
            error("operator[]-index out of order");
        return array[x];
      }
int dynarray::count(){
    int cnt = 0;
    for(int x=0; x<size;x++)
      if(array[x])
         cnt++;
     return cnt;
  }

struct leg_p{ //a legend peice
   char *leg_name;
   double num;
};

struct rule{ //a single pre rule structure
  char *pre;
```

```
  char *cln;
  double bel;
  double dis;
  double unc;
};

class legend : public dynarray{
 public:
   int add(leg_p *leg){return dynarray::add(leg);}
   void set_leg(char *anscif);
 };

static error_handler er("legend error:");

void legend::set_leg(char *anscif){
   leg_p *s[40];
   float da[40];
//   legend *temp;
   char ch[100];
//   temp = new legend;
   FILE *afp;
   if((afp = fopen(anscif, "r")) == NULL)
     er.terminate("can't open ansci file %s\n", anscif);
   int k=0, n=140;
   int i=0, flag=0;
   char *ss;
//   float *dd;
   char c;
   ss = new char;
   i=0;
   flag = 0;
   do {
    for(i=0; i<100; i++)
        ch[i]='\0';
    i=0;
    while((c = fgetc(afp)) != ':'){
        ch[i] = c;
        i++;
        if(i>99)
         er.terminate("legend name is longer than 100 characters");
      };
    s[k] = new leg_p;
    ss = realloc(ss, strlen(ch));
    strcpy(ss, ch);
    if((strcmp(ss, "end")) != 0){
    fscanf(afp, "%g", &da[k]);
    s[k]->leg_name = malloc(strlen(ss));
    strcpy(s[k]->leg_name, ss);
    s[k]->num = da[k];
//    printf("name %s num %g\n", s[k]->leg_name, s[k]->num);
    add(s[k]);
    k++;
    if(k>39)
      er.terminate("more than 40 legends, modify program!");
    while((c = fgetc(afp)) != '\n'){;}
  }
    else
      flag = 1;
  }while(flag == 0);
   delete s;
```

```
    delete da;
    fclose(afp);
  }

static error_handler erro("rule setup error: ");

class rules : public dynarray{
 public:
  int add(rule *rul){return dynarray::add(rul);}
  void set_rules(char *rulef);
};

void rules::set_rules(char *rulef){
    rule *s[50];
    float bel[50],dis[50],unc[50];
//    rules *temp;
    char *ss, *tt;
    FILE *rfp;
    int k=0, flag;
    char ch[100], ci[100];
    int i = 0;
    char c;
    ss = malloc(sizeof(char));
    tt = malloc(sizeof(char));
//    temp = new rules;
    if((rfp = fopen(rulef, "r")) == NULL)
      erro.terminate("can't open ruleb file %s\n", rulef);
    i=0;
    k=0;
    flag = 0;
    do {
      for(i=0; i<100; i++){
        ch[i] = '\0';
        ci[i] = '\0';
      }
      i = 0;
      while((c = fgetc(rfp)) != ','){
         ch[i] = c;
         i++;
         if(i>99)
           erro.terminate("pre longer than 100");
       }
     ss = realloc(ss, strlen(ch)+1);
     strcpy(ss, ch);
     if((strcmp(ss, "end")) !=0){
     i = 0;
     while((c = fgetc(rfp)) != '\n'){;}
     while((c = fgetc(rfp)) != '.'){
         ci[i] = c;
         i++;
         if(i>99)
           erro.terminate("cln longer than 100");
       }
     fscanf(rfp, "%g", &bel[k]);
     fscanf(rfp, "%g", &dis[k]);
     fscanf(rfp, "%g", &unc[k]);
     s[k] = new rule;
     tt = realloc(tt, strlen(ci)+1);
     strcpy(tt, ci);
     s[k]->pre = malloc(strlen(ss)+1);
     s[k]->cln = malloc(strlen(tt)+1);
```

```
      strcpy(s[k]->pre, ss);
      strcpy(s[k]->cln, tt);
      s[k]->bel = bel[k];
      s[k]->dis = dis[k];
      s[k]->unc = unc[k];
      add(s[k]);
//      printf("if %s\nthen %s\n", s[k]->pre, s[k]->cln);
//      printf("%g %g %g\n", s[k]->bel, s[k]->dis, s[k]->unc);
      k++;
      if(k>49)
         erro.terminate("rules more than 50");
      while((c = fgetc(rfp)) != '\n'){;}
    }
      else
       flag = 1;
     }while(flag == 0);
//    printf("i=%d k=%d", i, k);
     fclose(rfp);
//    return temp;
     delete s;
     delete bel;
     delete dis;
     delete unc;
  }

//class for numeric input

static error_handler ern("nrule error: ");

struct nrule{
  char *pre; //can be greater, less_eq, between.
  char *cln;
  double num1, num2; //interval
  double bel, dis, unc; //belief function of the rule
};

class num_rul : public dynarray{
 public:
  int add(nrule *nr){return dynarray::add(nr);}
  void set_nr(char *nrf);
};

void num_rul::set_nr(char *nrf){
    nrule *s[40];
    float b[40], d[40], u[40], n1[40], n2[40];
    char *ss, *tt;
    FILE *rfp;
    int k=0, flag;
    char c, ch[100], ci[100];
    int i=0;

    ss = malloc(sizeof(char));
    tt = malloc(sizeof(char));
    if ((rfp = fopen(nrf, "r")) == NULL)
      ern.terminate("can't open nrule file %s for input\n", nrf);
    i = 0;
    k = 0;
    flag = 0;
    do {
      for (i=0; i<100; i++){
        ch[i] = '\0';
        ci[i] = '\0';
```

```
      }
      i = 0;
      while ((c = fgetc(rfp)) != ','){
          ch[i] = c;
          i++;
          if (i > 99)
            ern.terminate("if longer than 100");
}
       ss = realloc(ss, strlen(ch)+1);
       strcpy(ss, ch);
       n2[k] = 0;
       if((strcmp(ss, "end")) !=0){
        if ((strcmp(ss, "greater_than")) == 0 ||
            (strcmp(ss, "less_than")) == 0){
           fscanf(rfp, "%g", &n1[k]);
 }
        else if
           ((strcmp(ss, "between")) == 0){
             fscanf(rfp, "%g", &n1[k]);
             fscanf(rfp, "%g", &n2[k]);
   }
        else
          ern.terminate("Sorry, I don't know %s\n", ss);
       i = 0;
       while((c = fgetc(rfp)) != '\n'){;}
       while((c = fgetc(rfp)) != '.'){
      ci[i] = c;
        i++;
      if(i>99)
        ern.terminate("then longer than 100");
 }
        fscanf(rfp, "%g", &b[k]);
        fscanf(rfp, "%g", &d[k]);
fscanf(rfp, "%g", &u[k]);
        s[k] = new nrule;
tt = realloc(tt, strlen(ci)+1);
strcpy(tt, ci);
s[k]->pre = malloc(strlen(ss)+1);
s[k]->cln = malloc(strlen(tt)+1);
strcpy(s[k]->pre, ss);
strcpy(s[k]->cln, tt);
        s[k]->bel = b[k];
        s[k]->dis = d[k];
        s[k]->unc = u[k];
        s[k]->num1 = n1[k];
        s[k]->num2 = n2[k];
        add(s[k]);
//    printf("if %s\nthen %s\n", s[k]->pre, s[k]->cln);
//    printf("%g %g %g\n", s[k]->bel, s[k]->dis, s[k]->unc);
        k++;
if(k>39)
         ern.terminate("nrules more than 40");
        while((c = fgetc(rfp)) != '\n'){;}
      }
      else
        flag = 1;
   }while(flag == 0);
   fclose(rfp);
   delete s;
   delete b;
```

```
    delete d;
    delete u;
  }

//**************** end of util.h ****************************//

// network.h --- defines the network.

struct bfn{
   double bel;
   double dis;
   double unc;
 };

struct conn{ //a connection to other node
   char *towho; //name this connection points to
   char *status; //can be yes or no. default is no.
   char *where; //connect to up or down
   char *how; //relation with other conn. canbe and or single.
   char *withwho;//name of an other conn
   char *flag; //visited or not by searcher
 };

static error_handler ere("network error:");

class node : public dynarray{ //piece of node
 public:
   int add(conn *con){return dynarray::add(con);}
    node *set_nd(FILE *, int);
 };


node *node::set_nd(FILE * nodeptr, int d){
    char ch[40];
    conn *u[20];
    char c;
    char *yes, *no, *unvstd;
    yes = "yes";
    no = "no";
    unvstd = "unvisited";
    int i = 0;
    node *temp = new node;
  for(int j=0; j<d; j++){
//    printf("j= %d\n", j);
//    u[j] = (conn*)malloc(sizeof(conn));
    u[j] = new conn;
    for(i=0; i<40; i++)
        ch[i] = '\0';
    i = 0;
//    while((c = fgetc(nodeptr)) != '\n'){;}
    while((c = fgetc(nodeptr)) != ','){
          ch[i] = c;
          i++;
}
//    st = realloc(st, strlen(ch)+1);
//    strcpy(st, ch);
    u[j]->towho = malloc(sizeof(ch)+1);
    strcpy(u[j]->towho, ch);
    for(i=0; i<40; i++)
        ch[i] = '\0';
    i = 0;
```

```
    while((c = fgetc(nodeptr)) != ','){
          ch[i] = c;
          i++;
}
//    st = realloc(st, strlen(ch)+1);
//    strcpy(st, ch);
    u[j]->where = malloc(sizeof(ch)+1);
    strcpy(u[j]->where, ch);
    for(i=0; i<40; i++)
         ch[i] = '\0';
    i = 0;
    while((c = fgetc(nodeptr)) != ','){
        ch[i] = c;
        i++;
      }
//     st = realloc(st, strlen(ch)+1);
//     strcpy(st, ch);
     u[j]->how = malloc(sizeof(ch)+1);
     strcpy(u[j]->how, ch);
     for(i=0; i<40; i++)
          ch[i] = '\0';
     i = 0;
     while((c = fgetc(nodeptr)) != ','){
          ch[i] = c;
          i++;
 }
//     st = realloc(st, strlen(ch)+1);
//     strcpy(st, ch);
     u[j]->withwho = malloc(sizeof(ch)+1);
     strcpy(u[j]->withwho, ch);
     u[j]->flag = malloc(strlen(unvstd)+1);
     strcpy(u[j]->flag, unvstd);
     u[j]->status = malloc(strlen(no)+1);
     strcpy(u[j]->status, no);
//       u[j]->flag = "unvisited";
//       u[j]->status = "no";
     temp->add(u[j]);
     while ((c = fgetc(nodeptr)) != '\n'){;}
  }
   return temp;
}

struct nd{
   char *ndname;
   node *nds;
 };

/*class netwk : public dynarray{
 public:
   netwk *set_nwk(char *netf);
   int add(nd *nod){return dynarray::add(nod);}
 };
*/
class network : public dynarray{
   char *netname;
//   netwk *target;
 public:
   network(char *netf, char *netn);
   ~network();
   void set_nwk(char *netf);
```

```
    int add(nd *nod){return dynarray::add(nod);}
    nd *getnd(char *);
    nd *getnd(int);
    void set_status(char *, char *);// first arg. must be
       // a terminal node and 2 args. must be same.
    void write_netwk();
    char *get_towhoup(char *);//get up towho node
    int terminalp(char *);
    int set_yes(char *, char *);
    int yesp(char *, char *);
    void write_node(nd *);
    void write_conn(conn *);
    int visitedp(char *, char*);
    int unvisitedp(char *);
    int tmnl_or_ndwn(char *);
    int topp(char *);//return 1 if true
    char *pick_towho(char *);//pick out a unvisited, yes and
        //down conn.
    char *get_towho(char *, char *);//get towho of next conn
        // 1st arg. is ndnm and 2nd is withwho of last conn
    int set_visited(char *, char *);
    char *get_withwho(char *, char *);// first arg. is ndnm and
       //second is towho of the conn.
    char *get_how(char *, char*); //first arg. is ndnm 2nd is towho
  };

 network::~network(){
   delete netname;
 }

 network::network(char *netf, char *netn){
    netname = malloc(strlen(netn)+1);
    strcpy(netname, netn);
    set_nwk(netf);
//    printf("net work set up\n");
    int n = count();
//    printf("target has %d nodes\n", n);
  }

void network::set_nwk(char *netf){
  char ch[40];
  char c;
  char *s, *st;
  st = new char;
  s = new char;
  int i, d, flag;
  FILE *netptr;
  if((netptr = fopen(netf, "r")) == NULL)
     ere.terminate("can't open %s for Input\n", netf);
//  while(feof(netptr) == 0){
  flag = 0;
 do{
   for(i=0; i<40; i++)
       ch[i] = '\0';
   i = 0;
   while((c = fgetc(netptr)) != ':'){
     ch[i] = c;
       i++;
   }
   fscanf(netptr, "%d", &d);
```

```
    while ((c = fgetc(netptr)) != '\n'){;}
    s = realloc(s, strlen(ch)+1);
    strcpy(s, ch);
    if((strcmp(s, "end")) == 0)
        flag = 1;
    else {
      if((strcmp(s, "susceptibility")) == 0){
       nd *susceptibility = new nd;
       st = "susceptibility";
       susceptibility->nds =
            susceptibility->nds->set_nd(netptr, d);
       susceptibility->ndname = malloc(strlen(st)+1);
       strcpy(susceptibility->ndname, st);
       add(susceptibility);
       }
      else if
        ((strcmp(s, "air_magnet")) == 0){
          nd *air_magnet = new nd;
          st = "air_magnet";
          air_magnet->nds = air_magnet->nds->set_nd(netptr, d);
          air_magnet->ndname = malloc(strlen(st)+1);
          strcpy(air_magnet->ndname, st);
          add(air_magnet);
        }
      else if
        ((strcmp(s, "image1")) == 0){
          nd *image1 = new nd;
          st = "image1";
          image1->nds = image1->nds->set_nd(netptr, d);
          image1->ndname = malloc(strlen(st)+1);
          strcpy(image1->ndname, st);
          add(image1);
        }
      else if
        ((strcmp(s, "image2")) == 0){
          nd *image2 = new nd;
          st = "image2";
          image2->nds = image2->nds->set_nd(netptr, d);
          image2->ndname = malloc(strlen(st)+1);
          strcpy(image2->ndname, st);
          add(image2);
        }
      else if
        ((strcmp(s, "ground_mag")) == 0){
          nd *ground_mag = new nd;
          st = "ground_mag";
          ground_mag->nds =
                ground_mag->nds->set_nd(netptr, d);
          ground_mag->ndname = malloc(strlen(st)+1);
          strcpy(ground_mag->ndname, st);
          add(ground_mag);
//          printf("ground_mag setup");
        }
      else if
        ((strcmp(s, "conductor")) == 0){
          nd *conductor = new nd;
          st = "conductor";
          conductor->nds = conductor->nds->set_nd(netptr, d);
          conductor->ndname = malloc(strlen(st)+1);
```

```
        strcpy(conductor->ndname, st);
        add(conductor);
      }
    else if
      ((strcmp(s, "high_con")) == 0){
        nd *high_con = new nd;
        st = "high_con";
        high_con->nds = high_con->nds->set_nd(netptr, d);
        high_con->ndname = malloc(strlen(st)+1);
        strcpy(high_con->ndname, st);
        add(high_con);
      }
    else if
      ((strcmp(s, "high_airmag")) == 0){
        nd *high_airmag = new nd;
        st = "high_airmag";
        high_airmag->nds = high_airmag->nds->set_nd(netptr, d);
        high_airmag->ndname = malloc(strlen(st)+1);
        strcpy(high_airmag->ndname, st);
        add(high_airmag);
      }
    else if
      ((strcmp(s, "vlf")) == 0){
        nd *vlf = new nd;
        st = "vlf";
        vlf->nds = vlf->nds->set_nd(netptr, d);
        vlf->ndname = malloc(strlen(st)+1);
        strcpy(vlf->ndname, st);
        add(vlf);
      }
    else if
      ((strcmp(s, "image")) == 0){
        nd *image = new nd;
        st = "image";
        image->nds = image->nds->set_nd(netptr, d);
        image->ndname = malloc(strlen(st)+1);
        strcpy(image->ndname, st);
        add(image);
      }
    else if
      ((strcmp(s, "vlf1")) == 0){
         nd *vlf1 = new nd;
         st = "vlf1";
         vlf1->nds = vlf1->nds->set_nd(netptr, d);
         vlf1->ndname = malloc(strlen(st)+1);
         strcpy(vlf1->ndname, st);
         add(vlf1);
}
    else if
      ((strcmp(s, "vlf2")) == 0){
         nd *vlf2 = new nd;
         st = "vlf2";
         vlf2->nds = vlf2->nds->set_nd(netptr, d);
         vlf2->ndname = malloc(strlen(st)+1);
         strcpy(vlf2->ndname, st);
         add(vlf2);
       }
    else if
      ((strcmp(s, "air_em")) == 0){
          nd *air_em = new nd;
```

```
            st = "air_em";
            air_em->nds = air_em->nds->set_nd(netptr, d);
            air_em->ndname = malloc(strlen(st)+1);
            strcpy(air_em->ndname, st);
            add(air_em);
}
    else if
      ((strcmp(s, "resistivity")) == 0){
            nd *resistivity = new nd;
            st = "resistivity";
            resistivity->nds = resistivity->nds->set_nd(netptr, d);
            resistivity->ndname = malloc(strlen(st)+1);
            strcpy(resistivity->ndname, st);
            add(resistivity);
}
    else if
      ((strcmp(s, "em_grade")) == 0){
            nd *em_grade = new nd;
            st = "em_grade";
            em_grade->nds = em_grade->nds->set_nd(netptr, d);
            em_grade->ndname = malloc(strlen(st)+1);
            strcpy(em_grade->ndname, st);
            add(em_grade);
}
    else if
      ((strcmp(s, "Input")) == 0){
            nd *Input = new nd;
            st = "Input";
            Input->nds = Input->nds->set_nd(netptr, d);
            Input->ndname = malloc(strlen(st)+1);
            strcpy(Input->ndname, st);
            add(Input);
}
    else if
      ((strcmp(s, "lake")) == 0){
            nd *lake = new nd;
            st = "lake";
            lake->nds = lake->nds->set_nd(netptr, d);
            lake->ndname = malloc(strlen(st)+1);
            strcpy(lake->ndname, st);
            add(lake);
}
    else if
      ((strcmp(s, "ground_em")) == 0){
            nd *ground_em = new nd;
            st = "ground_em";
            ground_em->nds = ground_em->nds->set_nd(netptr, d);
            ground_em->ndname = malloc(strlen(st)+1);
            strcpy(ground_em->ndname, st);
            add(ground_em);
}
    else if
      ((strcmp(s, "geology")) == 0){
            nd *geology = new nd;
            st = "geology";
            geology->nds = geology->nds->set_nd(netptr, d);
            geology->ndname = malloc(strlen(st)+1);
            strcpy(geology->ndname, st);
            add(geology);
```

```
   }
      else if
        ((strcmp(s, "density")) == 0){
             nd *density = new nd;
             st = "density";
             density->nds = density->nds->set_nd(netptr, d);
             density->ndname = malloc(strlen(st)+1);
             strcpy(density->ndname, st);
             add(density);
   }
      else if
        ((strcmp(s, "gravity")) == 0){
            nd *gravity = new nd;
            st = "gravity";
            gravity->nds = gravity->nds->set_nd(netptr, d);
            gravity->ndname = malloc(strlen(st)+1);
            strcpy(gravity->ndname, st);
            add(gravity);
 }
      else if
        ((strcmp(s, "target")) == 0){
           nd *target = new nd;
           st = "target";
           target->nds = target->nds->set_nd(netptr, d);
           target->ndname = malloc(strlen(st)+1);
           strcpy(target->ndname, st);
           add(target);
}
      else if
        ((strcmp(s, "chargeability")) == 0){
           nd *chargeability = new nd;
           st = "chargeability";
           chargeability->nds =
                chargeability->nds->set_nd(netptr, d);
           chargeability->ndname = malloc(strlen(st)+1);
           strcpy(chargeability->ndname, st);
           add(chargeability);
}
      else if
        ((strcmp(s, "step_airmag")) == 0){
           nd *step_airmag = new nd;
           st = "step_airmag";
           step_airmag->nds =
               step_airmag->nds->set_nd(netptr, d);
           step_airmag->ndname = malloc(strlen(st)+1);
           strcpy(step_airmag->ndname, st);
           add(step_airmag);
}
      else if
        ((strcmp(s, "fav_geol")) == 0){
           nd *fav_geol = new nd;
           st = "fav_geol";
           fav_geol->nds =
                fav_geol->nds->set_nd(netptr, d);
           fav_geol->ndname = malloc(strlen(st)+1);
           strcpy(fav_geol->ndname, st);
           add(fav_geol);
}
      else{
```

```
            printf("network error: node %s unknown\n", s);
            exit(1);
          }
      }
      }while (flag != 1);
      fclose(netptr);
}

void network::write_netwk(){
        nd *p;
        conn *t;
        int ii = count();
        printf("network %s has %d node.\n", netname, ii);
        for(int i=0; i<ii; i++){
          p = getnd(i);
          int jj = p->nds->count();
          for(int j=0; j<jj; j++){
        t = (conn *)(*p->nds)[j];
        printf("ndmame=%s %s %s %s %s %s %s\n", p->ndname,
           t->towho, t->status, t->where, t->how, t->withwho, t->flag);
      }
    }
  }

nd *network::getnd(int i){
   nd *tp;
   tp = (nd *)take(i);
   return tp;
}

nd *network::getnd(char *ndnm){
     int cnt, i;
     nd *tp;
     cnt = count();
     i = 0;
     do {
       tp = (nd *)take(i);
       if((strcmp(tp->ndname, ndnm)) == 0)
           i = cnt;
       else{
           i++;
           if(i == cnt)
               ere.terminate("node %s not found\n", ndnm);
}
     }while(i<cnt);
     return tp;
 }

void network::write_node(nd *pt){
      conn *p;
      int jj = pt->nds->count();
      for(int j=0; j<jj; j++){
      p = (conn *)(*pt->nds)[j];
       printf("ndmame=%s %s %s %s %s %s %s\n", pt->ndname,
          p->towho, p->status, p->where, p->how, p->withwho, p->flag);
    }
 }

void network::write_conn(conn *p){
       printf("conn name=%s %s %s %s %s %s\n", p->towho,
```

```
            p->status, p->where, p->how, p->withwho, p->flag);
    }

int network::yesp(char *ndnm, char *cnm){
   nd *tp;
   conn *p;
   int flag;
   flag = 0;
   tp = getnd(ndnm);
   int n = tp->nds->count();
   int i = 0;
   do {
      p = (conn *)(*tp->nds)[i];
     if(((strcmp(p->status, "yes")) == 0) &&
          ((strcmp(p->towho, cnm)) == 0)){
         flag = 1;
         i = n;
       }
     else
        i++;
    } while(i<n);
//   printf("%d returned for yesp(%s, %s)\n",flag,ndnm,cnm);
   return flag;
}

int network::visitedp(char *ndnm, char *cnm){
   nd *tp;
   conn *p;
   int flag;
   flag = 0;
   tp = getnd(ndnm);
   int n = tp->nds->count();
   int i = 0;
   do {
      p = (conn *)(*tp->nds)[i];
     if(((strcmp(p->flag, "visited")) == 0) &&
        ((strcmp(p->towho, cnm)) == 0)){
         flag = 1;
         i = n;
       }
     else
        i++;
    } while(i<n);
//   printf("%d returned for visitedp(%s, %s)\n",flag,ndnm,cnm);
   return flag;
}

int network::unvisitedp(char *ndnm){
   nd *tp;
   conn *p;
   int flag;
   flag = 0;
   tp = getnd(ndnm);
   int n = tp->nds->count();
   int i = 0;
   do {
      p = (conn *)(*tp->nds)[i];
     if(((strcmp(p->flag, "unvisited")) == 0) &&
        ((strcmp(p->status, "yes")) == 0)){
         flag = 1;
         i = n;
       }
```

```
      else
         i++;
     } while(i<n);
//   printf("%d returned for unvisitedp(%s)\n",flag,ndnm);
    return flag;
}

int network::tmnl_or_ndwn(char *ndnm){
    int flag;
    flag = 0;
    if(((terminalp(ndnm)) == 1) || ((unvisitedp(ndnm)) == 0))
       flag = 1;
//     printf("%d returned for tmnl_or_ndwn(%s)\n",flag,ndnm);
     return flag;
 }

int network::topp(char* ndnm){//return 1 if true 0 otherwise
    nd *tp;
    conn *p;
    int flag;
    flag = 1;
    tp = getnd(ndnm);
    int n = tp->nds->count();
    int i = 0;
    do {
       p = (conn *)(*tp->nds)[i];
//      write_conn(p);
      if((strcmp(p->where, "up")) != 0)
          i++;
      else{
          flag = 0;
          i = n;
        }
     } while(i<n);
    return flag;
}

char *network::get_towhoup(char *ndnm){
    nd *tp;
    char *temp = "none";
    int i, flag;
    conn *p;
    tp = getnd(ndnm);
    int n = tp->nds->count();
    i = 0;
    flag = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if(strcmp(p->where, "up") == 0){
//        temp = realloc(temp, strlen(p->towho)+1);
//        strcpy(temp, p->towho);
      temp = p->towho;
        flag = 1;
        i = n;
      }
      else
        i++;
     } while(i<n);
     return temp;
 }
```

```
int network::terminalp(char *ndnm){
   nd *tp;
   conn *p;
   int flag;
   flag = 1;
   tp = getnd(ndnm);
   int n = tp->nds->count();
   int i = 0;
   do {
     p = (conn *)(*tp->nds)[i];
     if((strcmp(p->where, "down")) != 0)
         i++;
     else{
         flag = 0;
         i = n;
       }
    } while(i<n);
//   printf("%d returned for terminalp(%s)\n",flag,ndnm);
   return flag;
}

char *network::pick_towho(char *ndnm){
   nd *tp;
   conn *p;
   char *temp;
   temp = malloc(strlen("none")+1);
   strcpy(temp, "none");
   int i, n;
   tp = getnd(ndnm);
   n = tp->nds->count();
   i = 0;
   do {
     p = (conn *)(*tp->nds)[i];
     if(((strcmp(p->status, "yes")) == 0) &&
        ((strcmp(p->flag, "unvisited")) == 0) &&
        ((strcmp(p->where, "down")) == 0)){
       i = n;
       temp = realloc(temp, strlen(p->towho)+1);
       strcpy(temp, p->towho);
     }
   else
     i++;
   } while(i<n);
   return temp;
}

char *network::get_withwho(char *ndnm, char *towho){
    nd *tp;
    conn *p;
    char *temp;
    char *s = "none";
    int i, n;
    tp = getnd(ndnm);
    n = tp->nds->count();
    i = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if((strcmp(p->towho, towho)) == 0){
         temp = malloc(strlen(p->withwho)+1);
         strcpy(temp, p->withwho);
         i = n;
```

```
        }
        else{
          i++;
          if(i == n){
             temp = malloc(strlen(s)+1);
             strcpy(temp, s);
  }
        }
     } while(i<n);
//    printf("%s returned for get_withwho(%s,%s)\n",temp,ndnm,towho);
    return temp;
  }


int network::set_visited(char *ndnm, char *towho){
    nd *tp;
    conn *p;
    int i, n;
    char *s = "visited";
    int flag;
    tp = getnd(ndnm);
    n = tp->nds->count();
    i = 0;
    flag = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if((strcmp(p->towho, towho)) == 0){
      p->flag = realloc(p->flag, strlen(s)+1);
      strcpy(p->flag, s);
      flag = 1;
      i = n;
    }
     else
       i++;
    } while(i<n);
    return flag;
  }

char *network::get_how(char *ndnm, char *towho){
    nd *tp;
    conn *p;
    char *s = "none";
    char *temp;
    int i, n;
    int flag;
    tp = getnd(ndnm);
    n = tp->nds->count();
    i = 0;
    flag = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if((strcmp(p->towho, towho)) == 0){
         temp = malloc(strlen(p->how)+1);
         strcpy(temp, p->how);
         flag = 1;
         i = n;
       }
    else{
      i++;
      if(i == n){
        temp = malloc(strlen(s)+1);
```

```
        strcpy(temp, s);
      }
    }
    } while(i<n);
//   printf("%s returnd for get_how(%s, %s)\n",temp,ndnm,towho);
   return temp;
  }

char *network::get_towho(char *ndnm, char *withwho){
      // get next conn in the same node.
    nd *tp;
    conn *p;
    char * temp = "none";
    int i, n;
    int flag;
    tp = getnd(ndnm);
    n = tp->nds->count();
    i = 0;
    flag = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if(((strcmp(p->towho, withwho)) == 0) &&
         ((strcmp(p->flag, "unvisited")) == 0) &&
         ((strcmp(p->status, "yes")) == 0)){
         temp = realloc(temp, strlen(p->towho)+1);
         strcpy(temp, p->towho);
         flag = 1;
         i = n;
       }
    else
      i++;
    } while(i<n);
    return temp;
  }

int network::set_yes(char *ndnm, char *towho){
    nd *tp;
    conn *p;
    int i, n;
    int flag;
    char *s = "yes";
    tp = getnd(ndnm);
    n = tp->nds->count();
    i = 0;
    flag = 0;
    do {
      p = (conn *)(*tp->nds)[i];
      if((strcmp(p->towho, towho)) == 0){
        if((strcmp(p->status, s)) !=0){
//           p->status = realloc(p->status, strlen(s)+1);
//           strcpy(p->status, s);
         p->status = "yes";
 }
      flag = 1;
      i = n;
      }
      else
        i++;
    } while(i<n);
    return flag;
```

```
  }

void network::set_status(char *ndnm, char *fromdown){
   char *sp, *st;
   printf("ndnm=%s fromdown=%s\n", ndnm, fromdown);
   if(terminalp(ndnm) == 1){
       if(strcmp((sp = get_towhoup(ndnm)), "none") == 0)
          ere.terminate("%s towhoup not found", ndnm);
//      printf("%s up is %s\n", ndnm,sp);
      if((set_yes(ndnm, sp)) == 0)
         printf("set conn %s to %s fialed\n", ndnm, sp);
//      write_node(getnd(ndnm));
    }
  do {
    st = ndnm;
    ndnm = sp;
    sp = get_towhoup(ndnm);
    printf("new ndnm is %s, lower node is %s, up is %s\n", ndnm, st, sp);
    set_yes(ndnm, st);
//    write_node(getnd(ndnm));
    set_yes(ndnm, sp);
//    write_node(getnd(ndnm));
  } while((topp(sp)) == 0);
 set_yes(sp, ndnm);
 }

//********************** end of network.h ***************************//

// globe.h --- contains functions for globe use

static error_handler e("bfmatrix computation error:");

bfmatrix *and(bfmatrix *A, bfmatrix *B){
      int ar = A->gr();
      int ac = A->gc();
      int br = B->gr();
      int bc = B->gc();
   if(ar!=br || ac!=bc){
      printf("%d %d %d %d\n",ar, br,ac,bc);
      e.terminate("bfmatrix dimensions are not equal\n");
    }
   matrix *I = new matrix(ar, ac, 1);
   matrix *k = new matrix(ar, ac, 1);
   *k = (*I)/((*I)-((*(A->bel))*(*(B->dis)))
           -((*(A->dis))*(*(B->bel))));
   *(A->bel) = (*(A->bel))*(*(B->bel))*(*k);
   *(A->dis) = (*(A->dis))*(*(B->dis))*(*k);
   delete I;
   delete k;
//   delete B;
   (*B).bfmatrix::~bfmatrix();
   return A;
    };

bfmatrix *or(bfmatrix *A, bfmatrix *B){
      int ar = A->gr();
      int ac = A->gc();
      int br = B->gr();
      int bc = B->gc();
```

```
    if(ar!=br || ac!=bc){
       printf("%d %d %d %d\n",ar,br,ac,bc);
       e.terminate("bfmatrix dimensions are not equal\n");
     }
    matrix *I = new matrix(ar, ac, 1);
    matrix *k = new matrix(ar, ac, 1);
    *k = (*I)/((*I)-((*(A->bel))*(*(B->dis)))
              -((*(A->dis))*(*(B->bel))));
    *I = (*(A->bel)-((*(A->bel))*(*(B->dis)))+(*(B->bel))
               -((*(B->bel))*((*(A->bel))+(*(A->dis)))))*(*k);
    *(A->dis) = ((*(A->dis))-(*(A->dis)*(*(B->bel)))
              +(*(B->dis))-(((*(A->bel))+(*(A->dis)))*(*(B->dis))))*(*k);
    *(A->bel) = *I;
    delete I;
    delete k;
//    delete B;
    (*B).bfmatrix::~bfmatrix();
    return A;
     }

bfmatrix *pass_rule(bfmatrix *A, bfn *b){
       int ar = A->gr();
       int ac = A->gc();
    matrix *I = new matrix(ar, ac, 1);
    matrix *k = new matrix(ar, ac, 1);
    matrix *Bb = new matrix(ar,ac,b->bel);
    matrix *Bd = new matrix(ar,ac,b->dis);
    *k = (*I)/((*I)-(*(A->dis))*(*Bd));
    *(A->dis) = (*(A->bel)*(*Bd)+((*(A->dis))*(*Bb)))*(*k);
    *(A->bel) = (*(A->bel))*(*Bb)*(*k);
     delete I;
     delete k;
     delete Bb;
     delete Bd;
     return A;
     }

//************************ end of globe.h ****************************//

//target.h --- defines target object

class target{
     int row, col;
     char *ndnm;
  public:
     bfmatrix *B;
     target(char *ndname, network &nw, int r, int c);
     ~target();
     void write_bfmap();
};

    target::target(char *ndname, network &nw, int r, int c){
    ndnm = malloc(strlen(ndname)+1);
    strcpy(ndnm, ndname);
    row = r; col = c;
    if((nw.topp(ndnm)) == 0){
        printf("%s is not a top node", ndnm);
        exit(1);
      }
 }

void target::write_bfmap(){
```

```
    B->print();
  }

target::~target(){
   delete ndnm;
//   delete B;
}

//************************ end of target.h ***********************//

// conductor.h ----- defines conductor object

class conductor{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    friend class target;
    conductor(char *ndname, char *rulef, network &, int r, int c);
    ~conductor();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   conductor::conductor(char *ndname, char *rulef, network &nw,
             int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("conductor rulebase set up successfully\n");
//   nw.write_netwk();
   int it = nw.terminalp("conductor");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("conductor");
   printf("form conductor up is to %s\n", p);
 }

rules *conductor::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
 }

void conductor::write_bfmap(){
    B->print();
  }

void conductor::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s\n then %s\n", f->pre, f->cln);
    printf("%g %g %g\n", f->bel, f->dis, f->unc);
    };
```

```
}

bfmatrix *conductor::apply_rules(char *upnd){
    rule *ru;
    int k, i, j;
    k = ruleb->count();
    bfn *belf = new bfn;
//    bfmatrix *temp = new bfmatrix(row, col);
    j = 0;
    for(i=0; i<k; i++){
    ru = (rule *)(*ruleb)[i];
    if((strcmp(upnd, ru->cln)) == 0){
         belf->bel = ru->bel;
         belf->dis = ru->dis;
         belf->unc = ru->unc;
      }
     else
       j++;
     }
     if(j > k){
       printf("fav_geol error: Sorry, I don't know %s\n", upnd);
       exit(1);
      }
      else
        return (pass_rule(B, belf));
//      return temp;
 }

conductor::~conductor(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//****************** end of conductor.h *************************//

//fav_geol.h --- defines favorabe geological condition

class fav_geol{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    friend class target;
    fav_geol(char *ndname, char *rulef, network &nw, int r, int c);
    ~fav_geol();
    bfmatrix *apply_rules(char *ndnm);
    rules *rles(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   fav_geol::fav_geol(char *ndname, char *rulef, network &nw,
              int r, int c){
   ndnm = malloc(strlen(ndname)+1);
   strcpy(ndnm, ndname);
   row = r; col = c;
   printf("ndnm = %s row = %d col = %d\n", ndnm, row, col);
   ruleb = rles(rulef);
   write_rules("fav_geol rules");
```

```
    printf("fav_geol rulebase set up successfully\n");
    int it = nw.terminalp("fav_geol");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("fav_geol");
    printf("from fav_geol up is to %s\n", p);
 }

rules *fav_geol::rles(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

void fav_geol::write_bfmap(){
    B->print();
  }

void fav_geol::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
           f->bel, f->dis, f->unc);
    }
}

bfmatrix *fav_geol::apply_rules(char *upndnm){
     rule *ru;
    int k, i, j;
    k = ruleb->count();
    bfn *belf = new bfn;
//    bfmatrix *temp = new bfmatrix(row, col);
    j = 0;
    for(i=0; i<k; i++){
      ru = (rule *)(*ruleb)[i];
      if((strcmp(upndnm, ru->cln)) == 0){
         belf->bel = ru->bel;
         belf->dis = ru->dis;
         belf->unc = ru->unc;
       }
       else
         j++;
    }
    if(j > k){
      printf("fav_geol error: Sorry, I don't know %s\n", upndnm);
      exit(1);
    }
    else
      return (pass_rule(B, belf));
//    return temp;
}

fav_geol::~fav_geol(){
   delete ndnm;
//   delete B;
   delete ruleb;
}
```

```
//************************** end of fav_geol.h ***************//

//air_em.h

class air_em{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    air_em(char *ndname, char *rulef, network &nw, int r, int c);
    ~air_em();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   air_em::air_em(char *ndname, char *rulef, network &nw,
              int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
//   printf("air_em rulebase set up successfully\n");
   int it = nw.terminalp("air_em");
//   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("air_em");
//   printf("from air_em up is to %s\n", p);
 }

bfmatrix *air_em::apply_rules(char *upnd){
     rule *ru;
     int k, i, j;
     k = ruleb->count();
     bfn *belf = new bfn;
//     bfmatrix *temp = new bfmatrix(row, col);
     j = 0;
     for(i=0; i<k; i++){
       ru = (rule *)(*ruleb)[i];
       if((strcmp(upnd, ru->cln)) == 0){
           belf->bel = ru->bel;
           belf->dis = ru->dis;
           belf->unc = ru->unc;
       }
       else
        j++;
       }
      if(j == k){
         printf("air_em error: Sorry, I don't know %s\n", upnd);
         exit(1);
       }
       else
         return (pass_rule(B, belf));
//         return temp;
 }

rules *air_em::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
```

```
 }

void air_em::write_bfmap(){
    B->print();
  }

void air_em::write_rules(char *msg){
   rule *f;
   if(*msg)
     printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                f->bel, f->dis, f->unc);
    }
}

air_em::~air_em(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//*************************** end of air_em.h ***************//

//groundem.h

static error_handler gem("ground EM error: ");

class ground_em{
   char *mapname;
   int col, row;
   char *datfile;
   char *ansfile;
//   matrix *A;
//   bfmatrix *B;
   legend *grndem;
   rules *ruleb;
   bfn prcn;
   double precision;
//   bfmatrix *B;
 public:
   friend class conductor;
   ground_em(char *map, char *datf, char *ansf,
                char *rulf, network &, int r, int r);
    ~ground_em();
    bfmatrix *apply_rules(char *);
    matrix *dat(char *);
    legend *lnd(char *);
    rules *rulb(char *);
    double get_precision(char *);
    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
    void write_bfmap();
};

   ground_em::ground_em(char *map, char *datf, char *ansf,
              char *rulf, network &nw, int r, int c){
   mapname = map;
```

```
    row = r; col = c;
    datfile = malloc(strlen(datf)+1);
    strcpy(datfile, datf);
    ansfile = malloc(strlen(ansf)+1);
    strcpy(ansfile, ansf);
    grndem = lnd(ansf);
    printf("ground_em legend set up successfully\n");
    ruleb = rulb(rulf);
    printf("ground_em rulebase set up successfully\n");
    nw.set_status("ground_em", "ground_em");
//    nw.write_netwk();
    int it = nw.terminalp("ground_em");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("ground_em");
    printf("form ground_em up is to %s\n", p);
 }

matrix *ground_em::dat(char *datf){
    matrix *Gr = new matrix(row, col);
    Gr->readat(datf);
    return Gr;
 }

legend *ground_em::lnd(char *ansf){
     legend *temp = new legend;
     temp->set_leg(ansf);
     return temp;
  }

rules *ground_em::rulb(char *rulf){
     rules *temp = new rules;
     temp->set_rules(rulf);
     return temp;
  }

double ground_em::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       print("If not, please enter your new number:");
```

```
   }
   }while(flag == 0);
  }
   prcn.bel = temp;
   prcn.dis = 0;
   prcn.unc = 1.0 - temp;
   return temp;
  }

void ground_em::write_legend(char *msg){
    leg_p *p;
    if(*msg) printf("%s\n", msg);
    int k = grndem->count();
   for(int i=0; i<k; i++){
     p = (leg_p *)(*grndem)[i];
      printf("%s  %g\n", p->leg_name, p->num);
   }
}

void ground_em::write_rules(char *msg){
    rule *f;
    if(*msg)
        printf("%s\n", msg);
    int k = ruleb->count();
    for(int i=0; i<k; i++){
     f = (rule *)(*ruleb)[i];
     printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                 f->bel, f->dis, f->unc);
     }
}

ground_em::~ground_em(){
    delete mapname;
//    delete A;
//    delete B;
//    delete prcn;
    delete ruleb;
    delete grndem;
}

//void ground_em::write_map(char *msg){
//    A->print(msg);
// }

bfmatrix *ground_em::apply_rules(char *upndnm){
    double legend_num[40];
    double b[40], d[40];
    int k, l, n, i, j, is;
    bfmatrix *B = new bfmatrix(row,col);
    rule *ru;
    leg_p *le;
    precision = get_precision("ground_em");
    matrix *A = dat(datfile);
    printf("ground_em data set up successfully\n");
    k = grndem->count();
    l = ruleb->count();
    n = 0;
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
```

```
        le = (leg_p *)(*grndem)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        };
      };
   };
   double dt;
   is = 0;
//   A->print("A");
   for(i=0; i<row; i++)
     for(j=0; j<col; j++){
         dt = (*A).val(i,j);
       for(int m=0; m<n; m++){
         if(dt == legend_num[m]){
//            printf("dt=%g ln=%g\n", dt, legend_num[m]);
           B->bfval(i,j,b[m],d[m]);
           is++;
  }
    }
   }
   if (is != row*col)
     gem.terminate("Please check if ground EM rules cover all legends!\n");
   delete A;
   return (pass_rule(B, &prcn));
 }

//********************** end of groundem.h ****************************//

//em_grade.h

static error_handler emg("grade air EM error: ");

class em_grade{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    em_grade(char *map_name, char *datf, char *anscif,
            char *rulef, network &, int r, int c);
    ~em_grade();
    bfmatrix *apply_rules(char *);
    matrix *dat(char *);
    legend *lnd(char *);
    rules *rulb(char *);
    double get_precision(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};
```

```
em_grade::em_grade(char *map_name, char *datf, char *anscif,
                   char *rulef, network &nw, int r, int c){
   mapname = map_name;
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lnd(anscif);
   printf("em_grade legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("em_grade rulebase set up successfully\n");
//   B = apply_rules(nw);
   nw.set_status("em_grade", "em_grade");
//   nw.write_netwk();
   int it = nw.terminalp("em_grade");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("em_grade");
   printf("from em_grade up is to %s\n", p);
 }

matrix *em_grade::dat(char *datf){
    matrix *Em = new matrix(row, col);
    Em->readat(datf);
    return Em;
  }

legend *em_grade::lnd(char *asnf){
   legend *temp = new legend;
   temp->set_leg(asnf);
   return temp;
 }

rules *em_grade::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
 }

double em_grade::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,", ndnm);
   printf("0 means nothing believeable on this map");
   printf("and 1 means you totally believe the map is true.\n");
   do {
   scanf("%lf", &temp);
   if(temp >1)
     printf("The number you given greater than 1.\nPlease reenter:");
   else if(temp < 0)
      printf("The number you entered less than 0.\nPlease reenter:");
   else{
    printf("Your certainty about %s map is %g\n", ndnm, temp);
    c = getchar();
```

```
    if(c == 'y')
       flag = 1;
    else
      printf("Is this number looks ok for you?(y/n)");
      printf("If not, please enter your new number:");
  }
  }while(flag == 0);
 }
  prcn.bel = temp;
  prcn.dis = 0;
  prcn.unc = 1.0 - temp;
  return temp;
 }


//void em_grade::write_map(char *msg){
//   A->print(msg);
// }

void em_grade::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void em_grade::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                   f->bel, f->dis, f->unc);
    }
}

em_grade::~em_grade(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}

bfmatrix *em_grade::apply_rules(char *upndnm){
   printf("em_grade::apply_rules(%s) called\n",upndnm);
   double legend_num[40];
   double b[40], d[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("air_em_grade");
   matrix *A = dat(datfile);
```

```
    printf("em_grade data set up successfully\n");
    k = lgnd->count();
    l = ruleb->count();
    n = 0;
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
         le = (leg_p *)(*lgnd)[j];
         if(((strcmp(le->leg_name, ru->pre)) == 0) &&
            ((strcmp(upndnm, ru->cln)) == 0)){
           legend_num[n] = le->num;
           b[n] = ru->bel;
           d[n] = ru->dis;
//           u[n] = ru->unc;
           n++;
         }
       }
    }
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//            printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
     }
    if (is != row*col)
      emg.terminate("Please check if airEM_grade rulebase cover all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
 }

//********************* end of em_grade.h *******************************//

//input.h

static error_handler inp("INPUT error: ");

class Input{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    Input(char *, char *, char *, char *, network &, int, int);
    ~Input();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
```

```
    rules *rulb(char *);
    matrix *dat(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   Input::Input(char *map_name, char *datf, char *anscif,
               char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("Input legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("Input rulebase set up successfully\n");
   nw.set_status("Input", "Input");
//   nw.write_netwk();
   int it = nw.terminalp("Input");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("Input");
   printf("form Input up is to %s\n", p);
 }

matrix *Input::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *Input::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *Input::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

double Input::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,",ndnm);
   printf("0 means nothing believeable on this map and");
   printf("1 means you totally believe the map is true.\n");
   do {
   scanf("%lf", &temp);
   if(temp >1)
```

```
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
   }
   }while(flag == 0);
  }
   prcn.bel = temp;
   prcn.dis = 0;
   prcn.unc = 1.0 - temp;
   return temp;
  }

//void Input::write_map(char *msg){
//    A->print(msg);
// }

void Input::write_legend(char *msg){
    leg_p *p;
    if(*msg) printf("%s\n", msg);
    int k = lgnd->count();
   for(int i=0; i<k; i++){
     p = (leg_p *)(*lgnd)[i];
      printf("%s  %g\n", p->leg_name, p->num);
   }
}

void Input::write_rules(char *msg){
    rule *f;
    if(*msg)
        printf("%s\n", msg);
    int k = ruleb->count();
    for(int i=0; i<k; i++){
     f = (rule *)(*ruleb)[i];
     printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                f->bel, f->dis, f->unc);
     }
}

Input::~Input(){
    delete mapname;
//    delete A;
//    delete B;
//    delete prcn;
    delete ruleb;
    delete lgnd;
}

bfmatrix *Input::apply_rules(char *upndnm){
    double legend_num[40];
    double b[40], d[40];// u[40];
    int k, l, n, i, j, is;
    bfmatrix *B = new bfmatrix(row,col);
```

```
    rule *ru;
    leg_p *le;
    precision = get_precision("Input");
    matrix *A = dat(datfile);
    printf("Input data set up successfully\n");
    k = lgnd->count();
    l = ruleb->count();
    n = 0;
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
         le = (leg_p *)(*lgnd)[j];
         if(((strcmp(le->leg_name, ru->pre)) == 0) &&
            ((strcmp(upndnm, ru->cln)) == 0)){
           legend_num[n] = le->num;
           b[n] = ru->bel;
           d[n] = ru->dis;
//           u[n] = ru->unc;
           n++;
         };
       };
    };
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//             printf("dt=%g ln=%g\n", dt, legend_num[m]);
             B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
     }
     if (is != row*col)
        inp.terminate("Please check if INPUT rules cover all legends!\n");
     delete A;
     return (pass_rule(B, &prcn));
 }

//******************** end of input.h ******************************//

// geol.h

static error_handler ger("geology error: ");

class geology{
     char *mapname;
     int row, col;
     char *datfile;
     char *ansfile;
//     matrix *Geomap;
//     bfmatrix *B;
     legend *geo;
     rules *ruleb;
     bfn prcn;
     double precision;
     void set_prcn();
  public:
```

```
    geology(char *, char *, char *, char *, network &, int, int);
     ~geology();
     char *get_name(){return mapname;}
     bfmatrix *apply_rules(char *);
     matrix *dat(char *datf);
     rules *rles(char *);
     legend *lgnd(char *);
     double get_precision(char *);
//     void write_map(char *msg);
     void write_legend(char *msg);
     void write_rules(char *msg);
};


    geology::geology(char *map_name, char *datf, char *anscif,
                char *rulef, network &nw, int r, int c){
    mapname = malloc(strlen(map_name));
    strcpy(mapname, map_name);
    row = r; col = c;
    datfile = malloc(strlen(datf)+1);
    strcpy(datfile, datf);
    ansfile = malloc(strlen(anscif)+1);
    strcpy(ansfile, anscif);
    geo = lgnd(anscif);
    ruleb = rles(rulef);
//   nw.write_netwk();
    nw.set_status("geology", "geology");
    int it = nw.terminalp("geology");
    char *p = nw.get_towhoup("geology");
 }

legend *geology::lgnd(char *ansf){
    legend *temp = new legend;
    temp->set_leg(ansf);
    return temp;
 }

rules *geology::rles(char *rulf){
    rules *rul = new rules;
    rul->set_rules(rulf);
    return rul;
 }

matrix *geology::dat(char *datf){
//    printf("Geomap(%d, %d) \n", row, col);
    matrix *Geo = new matrix(row, col);
    Geo->readat(datf);
    return Geo;
 }

double geology::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map");
```

```
    printf("and 1 means you totally believe the map is true.\n");

    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
   }
   }while(flag == 0);
  }
   return temp;
  }

void geology::set_prcn(){
   prcn.bel = precision;
   prcn.dis = 0;
   prcn.unc = 1.0 - precision;
}

//void geology::write_map(char *msg){
//   Geomap->print(msg);
// }

void geology::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = geo->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*geo)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void geology::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                f->bel, f->dis, f->unc);
    };
}

geology::~geology(){
   delete mapname;
//   delete Geomap;
//   delete B;
//   delete prcn;
   delete ruleb;
```

```
    delete geo;
}

bfmatrix *geology::apply_rules(char *upndnm){
    double legend_num[40];
    double b[40], d[40];// u[40];
    int k, l, n, i, j, is;
    bfmatrix *B = new bfmatrix(row,col);
    rule *ru;
    leg_p *le;
    precision = get_precision("geology");
    set_prcn();
    k = geo->count();
    l = ruleb->count();
    n = 0;
    matrix *Geomap = dat(datfile);
//    write_map("geolmap");
//  printf("geol data set up successfully\n");
//    write_rules("rules");
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
         le = (leg_p *)(*geo)[j];
         if(((strcmp(le->leg_name, ru->pre)) == 0) &&
            ((strcmp(upndnm, ru->cln)) == 0)){
           legend_num[n] = le->num;
           b[n] = ru->bel;
           d[n] = ru->dis;
//           u[n] = ru->unc;
           n++;
         }
       }
    }
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*Geomap).val(i,j);
        for(int m=0; m<n; m++){
//       printf("dt = %g legen_num = %g\n", dt, legend_num[m]);
         if(dt == legend_num[m]){
           B->bfval(i,j,b[m],d[m]);
            is++;
  }
     }
    }
    if (is != row*col)
     ger.terminate("Please check if geology subrulebase covers all legends.\n");
    delete Geomap;
    return (pass_rule(B, &prcn));
}

//********************** end of geol.h ******************************//

//ground_mag.h

static error_handler gndmg("ground_mag error: ");

class ground_mag{
```

```
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    ground_mag(char *, char *, char *, char *, network &, int, int);
    ~ground_mag();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   ground_mag::ground_mag(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("ground_mag legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("ground_mag rulebase set up successfully\n");
   nw.set_status("ground_mag", "ground_mag");
//   nw.write_netwk();
   int it = nw.terminalp("ground_mag");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("ground_mag");
   printf("from ground_mag up is to %s\n", p);
 }

matrix *ground_mag::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *ground_mag::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *ground_mag::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
```

```
    return temp;
  }

double ground_mag::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,",ndnm);
   printf("0 means nothing believeable on this map and");
   printf("1 means you totally believe the map is true.\n");
   do {
   scanf("%lf", &temp);
   if(temp >1)
     printf("The number you given greater than 1.\nPlease reenter:");
   else if(temp < 0)
      printf("The number you entered less than 0.\nPlease reenter:");
   else{
    printf("Your certainty about %s map is %g\n", ndnm, temp);
    c = getchar();
    if(c == 'y')
       flag = 1;
    else
      printf("Is this number looks ok for you?(y/n)");
      printf("If not, please enter your new number:");
  }
  }while(flag == 0);
 }
  prcn.bel = temp;
  prcn.dis = 0;
  prcn.unc = 1.0 - temp;
  return temp;
 }

void ground_mag::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void ground_mag::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

ground_mag::~ground_mag(){
```

```
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}

bfmatrix *ground_mag::apply_rules(char *upndnm){
   double legend_num[40];
   double b[40], d[40];// u[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("ground_mag");
   matrix *A = dat(datfile);
   printf("ground_mag data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        };
      };
   };
   for (i=0; i<n; i++)
      printf("%g %g %g\n", legend_num[i],b[i],d[i]);
   double dt;
   is = 0;
//   A->print("A");
   for(i=0; i<row; i++)
     for(j=0; j<col; j++){
         dt = (*A).val(i,j);
         l = is;
       for(int m=0; m<n; m++){
         if(dt == legend_num[m]){
//            printf("dt=%g ln=%g\n", dt, legend_num[m]);
           B->bfval(i,j,b[m],d[m]);
           is++;
  }
       }
     if (l == is)
       printf("%g ", dt);
    }
    if (is != row*col)
    gndmg.terminate("Pease check if ground_mag rules covers all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
}

//****************************** end of ground_mag.h ******************//
```

```
//air_magnet.h

static error_handler magerr("air_magnetic error: ");

class air_magnet{
    char *mapname;
    int row, col;
    char *datfile;
    char *nrulf;
//    matrix *magmap;
    num_rul *ruleb;
    bfn prcn;
//    bfmatrix *B;
    double precision;
    void set_prcn();
  public:
   air_magnet(char *, char *, char *, network &, int, int);
    ~air_magnet();
    char *get_name(){return mapname;}
    bfmatrix *apply_rules(char *);
    matrix *dat(char *datf);
    num_rul *rles(char *);
    double get_precision(char *);
//    void write_map(char *msg);
    void write_rules(char *msg);
};

//air_magnet::air_magnet(){;}

   air_magnet::air_magnet(char *map_name, char *datf,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   printf("air_magnet data size is %d * %d \n", row, col);
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   nrulf = malloc(strlen(rulef)+1);
   strcpy(nrulf, rulef);
   ruleb = rles(rulef);
   printf("air_mag rulebase set up successfully\n");
//   nw.write_netwk();
   nw.set_status("air_magnet", "air_magnet");
   int it = nw.terminalp("air_magnet");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("air_magnet");
   printf("from air_magnet up is to %s\n", p);
 }

num_rul *air_magnet::rles(char *rulf){
   num_rul *rul = new num_rul;
   rul->set_nr(rulf);
   return rul;
 }

matrix *air_magnet::dat(char *datf){
   printf("air_magnet dat setup size is %d * %d \n", row, col);
   matrix *Geo = new matrix(row, col);
   Geo->readat(datf);
```

```
    return Geo;
  }

 double air_magnet::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map");
    printf("and 1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
   }
   }while(flag == 0);
  }
   return temp;
  }

void air_magnet::set_prcn(){
   prcn.bel = precision;
   prcn.dis = 0;
   prcn.unc = 1.0-precision;
   printf("precision is set to belief = %g\n",precision);
   printf("prcn = %g %g %g\n",prcn.bel,
                prcn.dis, prcn.unc);
}

//void air_magnet::write_map(char *msg){
//    magmap->print(msg);
// }

void air_magnet::write_rules(char *msg){
   nrule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (nrule *)(*ruleb)[i];
    printf("if %s %g %g\n then %s (%g %g %g)\n", f->pre,
           f->num1, f->num2, f->cln, f->bel, f->dis, f->unc);
    };
}

air_magnet::~air_magnet(){
```

```
   delete mapname;
//   delete B;
   delete ruleb;
//   delete magmap;
//   delete prcn;
   delete datfile;
   delete nrulf;
}

bfmatrix *air_magnet::apply_rules(char *upndnm){
   char *pre[40];
   double n1[40], n2[40], b[40], d[40];// u[40];
   int l, n, i, j, k, is;
   bfmatrix *B = new bfmatrix(row,col);
   nrule *ru;
   precision = get_precision("air_magnet");
   set_prcn();
   printf("prcn = %g %g %g\n",prcn.bel,
               prcn.dis, prcn.unc);
   l = ruleb->count();
   n = 0;
   write_rules("air magnet rules");
   matrix *magmap = dat(datfile);
//   write_map("magmap");
//   for (i=0; i<10; i++){
//     printf("%10.7g", (*magmap).val(10,i));
//   }
   printf("magnet data set up successfully\n");
//   write_rules("rules");
   printf("upnode is %s\n", upndnm);
   for(i=0; i<l; i++){
     ru = (nrule *)(*ruleb)[i];
         if ((strcmp(upndnm, ru->cln)) == 0){
          pre[n] = malloc(strlen(ru->pre)+1);
          strcpy(pre[n], ru->pre);
          n1[n] = ru->num1;
          n2[n] = ru->num2;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        }
      }
   for (i=0; i<n; i++)
     printf("%s %g %g %g %g\n", pre[i],n1[i],n2[i],b[i],d[i]);
   is = 0;
   for (k=0; k<n; k++){
     printf("%g %g %g %g\n", n1[k],n2[k],b[k],d[k]);
     if ((strcmp(pre[k], "greater_than")) == 0){
      for(i=0; i<row; i++)
       for(j=0; j<col; j++){
         if ((*magmap).val(i,j) >= n1[k]){
            B->bfval(i,j,b[k],d[k]);
            is++;
 }
      }
     }
     else if
        ((strcmp(pre[k], "less_than")) == 0){
      for(i=0; i<row; i++)
```

```
        for(j=0; j<col; j++){
          if ((*magmap).val(i,j) <= n1[k]){
             B->bfval(i,j,b[k],d[k]);
             is++;
  }
        }
      }
       else if
          ((strcmp(pre[k], "between")) == 0){
        for(i=0; i<row; i++)
         for(j=0; j<col; j++){
          if (((*magmap).val(i,j) >= n1[k]) &&
              ((*magmap).val(i,j) <= n2[k])){
             B->bfval(i,j,b[k],d[k]);
             is++;
  }
        }
      }
      else
        magerr.terminate("Sorry, I don't know %s.\n", pre[k]);
     }
     if (is < row*col)
        magerr.terminate("Please check if rules cover all magnetic values!\n");
     for (i=0; i<n-1; i++)
        delete pre[i];
        delete n1;
        delete b;
        delete d;
        delete n2;
        delete magmap;
     return (pass_rule(B, &prcn));
 }

//********************* end of air_magnet.h *************************//

//charge.h --- handles chargeability data

static error_handler cha("chargeability error: ");

class chargeability{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    chargeability(char *, char *, char *, char *, network &, int, int);
    ~chargeability();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
```

```
};

    chargeability::chargeability(char *map_name, char *datf, char *anscif,
                char *rulef, network &nw, int r, int c){
    mapname = malloc(strlen(map_name)+1);
    strcpy(mapname, map_name);
    row = r; col = c;
    datfile = malloc(strlen(datf)+1);
    strcpy(datfile, datf);
    ansfile = malloc(strlen(anscif)+1);
    strcpy(ansfile, anscif);
    lgnd = lgd(anscif);
    printf("chargeability legend set up successfully\n");
    ruleb = rulb(rulef);
    printf("chargeability rulebase set up successfully\n");
    nw.set_status("chargeability", "chargeability");
//    nw.write_netwk();
    int it = nw.terminalp("chargeability");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("chargeability");
    printf("form chargeability up is to %s\n", p);
 }

matrix *chargeability::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *chargeability::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *chargeability::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

double chargeability::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,", ndnm);
   printf("0 means nothing believeable on this map and");
   printf("1 means you totally believe the map is true.\n");
   do {
   scanf("%lf", &temp);
   if(temp >1)
     printf("The number you given greater than 1.\nPlease reenter:");
   else if(temp < 0)
      printf("The number you entered less than 0.\nPlease reenter:");
   else{
    printf("Your certainty about %s map is %g\n", ndnm, temp);
```

```
      c = getchar();
      if(c == 'y')
         flag = 1;
      else
        printf("Is this number looks ok for you?(y/n)");
        printf("If not, please enter your new number:");
    }
    }while(flag == 0);
   }
    prcn.bel = temp;
    prcn.dis = 0;
    prcn.unc = 1.0 - temp;
    return temp;
   }

//void chargeability::write_map(char *msg){
//    A->print(msg);
// }

void chargeability::write_legend(char *msg){
    leg_p *p;
    if(*msg) printf("%s\n", msg);
    int k = lgnd->count();
   for(int i=0; i<k; i++){
     p = (leg_p *)(*lgnd)[i];
      printf("%s  %g\n", p->leg_name, p->num);
   }
}

void chargeability::write_rules(char *msg){
    rule *f;
    if(*msg)
        printf("%s\n", msg);
    int k = ruleb->count();
    for(int i=0; i<k; i++){
     f = (rule *)(*ruleb)[i];
     printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                f->bel, f->dis, f->unc);
     }
}

chargeability::~chargeability(){
    delete mapname;
//    delete A;
//    delete B;
//    delete prcn;
    delete ruleb;
    delete lgnd;
}

bfmatrix *chargeability::apply_rules(char *upndnm){
    double legend_num[40];
    double b[40], d[40];// u[40];
    int k, l, n, i, j, is;
    bfmatrix *B = new bfmatrix(row,col);
    rule *ru;
    leg_p *le;
    precision = get_precision("chargeability");
    matrix *A = dat(datfile);
    printf("chargeability data set up successfully\n");
```

```
    write_rules("charge rules");
    k = lgnd->count();
    l = ruleb->count();
    n = 0;
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
         le = (leg_p *)(*lgnd)[j];
         if(((strcmp(le->leg_name, ru->pre)) == 0) &&
            ((strcmp(upndnm, ru->cln)) == 0)){
           legend_num[n] = le->num;
           b[n] = ru->bel;
           d[n] = ru->dis;
//           u[n] = ru->unc;
           n++;
         };
       };
    };
    for (i=0; i<n; i++){
      printf("%g %g %g\n", legend_num[i],b[i],d[i]);
    }
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
          l = is;
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//              printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
       if (l == is)
          printf("%g ", dt);
        }
     if (is != row*col)
    cha.terminate("Please check if rules of chargeability cover all legends!\n");
     delete A;
     return (pass_rule(B, &prcn));
 }

//************************ end of charge.h ************************//

//resis.h ----- handles resistivity data

static error_handler rer("resistivity error: ");

class resistivity{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
```

```
    resistivity(char *, char *, char *, char *, network &, int, int);
    ~resistivity();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   resistivity::resistivity(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("resistivity legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("resistivity rulebase set up successfully\n");
   nw.set_status("resistivity", "resistivity");
//   nw.write_netwk();
   int it = nw.terminalp("resistivity");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("resistivity");
   printf("form resistivity up is to %s\n", p);
 }

matrix *resistivity::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *resistivity::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *resistivity::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

double resistivity::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,",ndnm);
```

```
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
   }
   }while(flag == 0);
  }
   prcn.bel = temp;
   prcn.dis = 0;
   prcn.unc = 1.0 - temp;
   return temp;
  }

//void resistivity::write_map(char *msg){
//   A->print(msg);
// }

void resistivity::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void resistivity::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

resistivity::~resistivity(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}
```

```
bfmatrix *resistivity::apply_rules(char *upndnm){
   double legend_num[40];
   double b[40], d[40];// u[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("resistivity");
   matrix *A = dat(datfile);
   printf("resistivity data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        };
      };
   };
   for (i=0; i<n; i++)
      printf("%g %g %g\n", legend_num[i],b[i],d[i]);
   double dt;
   is = 0;
//   A->print("A");
   for(i=0; i<row; i++)
     for(j=0; j<col; j++){
         dt = (*A).val(i,j);
         l = is;
       for(int m=0; m<n; m++){
         if(dt == legend_num[m]){
//            printf("dt=%g ln=%g\n", dt, legend_num[m]);
           B->bfval(i,j,b[m],d[m]);
           is++;
  }
       }
     if (l == is)
       printf("%g ", dt);
    }
    if (is != row*col)
    rer.terminate("Pease check if resistivity rules cover all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
}

//********************** end of resis.h *******************************//

//vlf.h

class vlf{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
```

```
    bfmatrix *B;
    vlf(char *ndname, char *rulef, network &nw, int r, int c);
    ~vlf();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   vlf::vlf(char *ndname, char *rulef, network &nw,
            int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("vlf rulebase set up successfully\n");
   int it = nw.terminalp("vlf");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("vlf");
   printf("from vlf up is to %s\n", p);
 }

bfmatrix *vlf::apply_rules(char *upnd){
     rule *ru;
     int k, i, j;
     k = ruleb->count();
     bfn *belf = new bfn;
//     bfmatrix *temp = new bfmatrix(row, col);
     j = 0;
     for(i=0; i<k; i++){
       ru = (rule *)(*ruleb)[i];
       if((strcmp(upnd, ru->cln)) == 0){
           belf->bel = ru->bel;
           belf->dis = ru->dis;
           belf->unc = ru->unc;
       }
       else
        j++;
       }
      if(j > k){
         printf("fav_geol error: Sorry, I don't know %s\\n", upnd);
         exit(1);
       }
       else
         return (pass_rule(B, belf));
//         return temp;
 }

rules *vlf::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
 }

void vlf::write_bfmap(){
    B->print();
  }

void vlf::write_rules(char *msg){
   rule *f;
   if(*msg)
```

```
      printf("%s\n", msg);
    int k = ruleb->count();
    for(int i=0; i<k; i++){
     f = (rule *)(*ruleb)[i];
     printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                  f->bel, f->dis, f->unc);
     }
}

vlf::~vlf(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//***************** end of vlf.h ****************************** //

// vlf1.h

static error_handler vf1("VLF1 error: ");

class vlf1{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    vlf1(char *, char *, char *, char *, network &, int, int);
    ~vlf1();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   vlf1::vlf1(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("vlf1 legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("vlf1 rulebase set up successfully\n");
   nw.set_status("vlf1", "vlf1");
//   nw.write_netwk();
   int it = nw.terminalp("vlf1");
```

```
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("vlf1");
    printf("form vlf1 up is to %s\n", p);
  }

matrix *vlf1::dat(char *datfile){
    matrix *Inp = new matrix(row, col);
    Inp->readat(datfile);
    return Inp;
  }

legend *vlf1::lgd(char *ansf){
    legend *temp = new legend;
    temp->set_leg(ansf);
    return temp;
  }

rules *vlf1::rulb(char *rulf){
     rules *temp = new rules;
     temp->set_rules(rulf);
     return temp;
   }

double vlf1::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
  }
   }while(flag == 0);
 }
  prcn.bel = temp;
  prcn.dis = 0;
  prcn.unc = 1.0 - temp;
  return temp;
 }

//void vlf1::write_map(char *msg){
//    A->print(msg);
// }
```

```
void vlf1::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void vlf1::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

vlf1::~vlf1(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}

bfmatrix *vlf1::apply_rules(char *upndnm){
   double legend_num[40];
   double b[40], d[40];// u[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("vlf1");
   matrix *A = dat(datfile);
   printf("vlf1 data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        };
      };
   };
   double dt;
```

```
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//             printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
     }
     if (is != row*col)
       vf1.terminate("Please check if vlf1 rules cover all legend!\n");
     delete A;
     return (pass_rule(B, &prcn));
 }

//********************** end of vlf1.h ***************************//

//vlf2.h

static error_handler vf2("VLF2 error: ");

class vlf2{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    vlf2(char *, char *, char *, char *, network &, int, int);
    ~vlf2();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   vlf2::vlf2(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("vlf2 legend set up successfully\n");
   ruleb = rulb(rulef);
```

```
   printf("vlf2 rulebase set up successfully\n");
   nw.set_status("vlf2", "vlf2");
//   nw.write_netwk();
   int it = nw.terminalp("vlf2");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("vlf2");
   printf("form vlf2 up is to %s\n", p);
 }

matrix *vlf2::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *vlf2::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *vlf2::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

double vlf2::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
   flag = 0;
   while(flag == 0){
   printf("Please enter a number between 0 to 1 which shows");
   printf("the certainty based on the accuracy of the %s map,",ndnm);
   printf("0 means nothing believeable on this map and");
   printf("1 means you totally believe the map is true.\n");
   do {
   scanf("%lf", &temp);
   if(temp >1)
     printf("The number you given greater than 1.\nPlease reenter:");
   else if(temp < 0)
      printf("The number you entered less than 0.\nPlease reenter:");
   else{
    printf("Your certainty about %s map is %g\n", ndnm, temp);
    c = getchar();
    if(c == 'y')
       flag = 1;
    else
      printf("Is this number looks ok for you?(y/n)");
      printf("If not, please enter your new number:");
 }
 }while(flag == 0);
}
 prcn.bel = temp;
 prcn.dis = 0;
 prcn.unc = 1.0 - temp;
 return temp;
}
```

```
//void vlf2::write_map(char *msg){
//    A->print(msg);
// }

void vlf2::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void vlf2::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
              f->bel, f->dis, f->unc);
    }
}

vlf2::~vlf2(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}

bfmatrix *vlf2::apply_rules(char *upndnm){
   double legend_num[40];
   double b[40], d[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("vlf2");
   matrix *A = dat(datfile);
   printf("vlf2 data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
```

```
        };
      };
    };
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//             printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
    }
    if (is != row*col)
        vf2.terminate("Please check if vlf2 rules cover all legend!\n");
    delete A;
    return (pass_rule(B, &prcn));
 }

//************************* end of vlf2.h ************************ //

//suscep.h//LAKE class

static error_handler ler("lake error: ");

class lake{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    lake(char *, char *, char *, char *, network &, int, int);
    ~lake();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   lake::lake(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
```

```
    strcpy(ansfile, anscif);
    lgnd = lgd(anscif);
    printf("Input legend set up successfully\n");
    ruleb = rulb(rulef);
    printf("Input rulebase set up successfully\n");
    nw.set_status("lake", "lake");
//    nw.write_netwk();
    int it = nw.terminalp("lake");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("lake");
    printf("form Input up is to %s\n", p);
 }

matrix *lake::dat(char *datfile){
    matrix *Inp = new matrix(row, col);
    Inp->readat(datfile);
    return Inp;
 }

legend *lake::lgd(char *ansf){
    legend *temp = new legend;
    temp->set_leg(ansf);
    return temp;
 }

rules *lake::rulb(char *rulf){
     rules *temp = new rules;
     temp->set_rules(rulf);
     return temp;
  }

double lake::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
 }
 }while(flag == 0);
}
 prcn.bel = temp;
```

```
  prcn.dis = 0;
  prcn.unc = 1.0 - temp;
  return temp;
 }

void lake::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void lake::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

lake::~lake(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
}

bfmatrix *lake::apply_rules(char *upndnm){
   double legend_num[10];
   double b[10], d[10];// u[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("lake");
   matrix *A = dat(datfile);
   printf("lake data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
```

```
        };
      };
    };
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
           l = is;
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//              printf("dt=%g ln=%g\n", dt, legend_num[m]);
              B->bfval(i,j,b[m],d[m]);
            is++;
  }
     }
      if (l == is)
          printf("%g ", dt);
    }
    if (is != row*col)
       ler.terminate("Please check if lake rules cover all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
 }


class susceptibility{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    friend class target;
    susceptibility(char *ndname, char *rulef, network &, int r, int c);
    ~susceptibility();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   susceptibility::susceptibility(char *ndname, char *rulef, network &nw,
              int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("susceptibility rulebase set up successfully\n");
//    nw.write_netwk();
   int it = nw.terminalp("susceptibility");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("susceptibility");
   printf("form susceptibility up is to %s\n", p);
 }

rules *susceptibility::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
```

```
 }

void susceptibility::write_bfmap(){
    B->print();
  }

void susceptibility::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s\n then %s\n", f->pre, f->cln);
    printf("%g %g %g\n", f->bel, f->dis, f->unc);
    };
}

bfmatrix *susceptibility::apply_rules(char *upnd){
    rule *ru;
    int k, i, j;
    k = ruleb->count();
    bfn *belf = new bfn;
//    bfmatrix *temp = new bfmatrix(row, col);
    j = 0;
    for(i=0; i<k; i++){
    ru = (rule *)(*ruleb)[i];
    if((strcmp(upnd, ru->cln)) == 0){
         belf->bel = ru->bel;
         belf->dis = ru->dis;
         belf->unc = ru->unc;
      }
     else
       j++;
     }
     if(j > k){
       printf("fav_geol error: Sorry, I don't know %s\n", upnd);
       exit(1);
      }
      else
      return (pass_rule(B, belf));
//       return temp;
 }

susceptibility::~susceptibility(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//********************** end of suscep.h ************************//

//lake.h

static error_handler ler("lake error: ");

class lake{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
```

```
//    bfmatrix *B;
    legend *lgnd;
    rules *ruleb;
    bfn prcn;
    double precision;
  public:
    lake(char *, char *, char *, char *, network &, int, int);
    ~lake();
    bfmatrix *apply_rules(char *);
    double get_precision(char *);
    legend *lgd(char *);
    rules *rulb(char *);
    matrix *dat(char *);
//    void write_map(char *msg);
    void write_legend(char *msg);
    void write_rules(char *msg);
};

   lake::lake(char *map_name, char *datf, char *anscif,
              char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   ansfile = malloc(strlen(anscif)+1);
   strcpy(ansfile, anscif);
   lgnd = lgd(anscif);
   printf("Input legend set up successfully\n");
   ruleb = rulb(rulef);
   printf("Input rulebase set up successfully\n");
   nw.set_status("lake", "lake");
//   nw.write_netwk();
   int it = nw.terminalp("lake");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("lake");
   printf("form Input up is to %s\n", p);
 }

matrix *lake::dat(char *datfile){
   matrix *Inp = new matrix(row, col);
   Inp->readat(datfile);
   return Inp;
 }

legend *lake::lgd(char *ansf){
   legend *temp = new legend;
   temp->set_leg(ansf);
   return temp;
 }

rules *lake::rulb(char *rulf){
    rules *temp = new rules;
    temp->set_rules(rulf);
    return temp;
  }

double lake::get_precision(char *ndnm){
   double temp;
   int flag;
```

```
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
   }
   }while(flag == 0);
  }
   prcn.bel = temp;
   prcn.dis = 0;
   prcn.unc = 1.0 - temp;
   return temp;
  }

void lake::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void lake::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

lake::~lake(){
   delete mapname;
//   delete A;
//   delete B;
//   delete prcn;
   delete ruleb;
   delete lgnd;
```

```
}

bfmatrix *lake::apply_rules(char *upndnm){
   double legend_num[10];
   double b[10], d[10];// u[40];
   int k, l, n, i, j, is;
   bfmatrix *B = new bfmatrix(row,col);
   rule *ru;
   leg_p *le;
   precision = get_precision("lake");
   matrix *A = dat(datfile);
   printf("lake data set up successfully\n");
   k = lgnd->count();
   l = ruleb->count();
   n = 0;
   for(i=0; i<l; i++){
     ru = (rule *)(*ruleb)[i];
     for(j=0; j<k; j++){
        le = (leg_p *)(*lgnd)[j];
        if(((strcmp(le->leg_name, ru->pre)) == 0) &&
           ((strcmp(upndnm, ru->cln)) == 0)){
          legend_num[n] = le->num;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        };
      };
   };
   double dt;
   is = 0;
//   A->print("A");
   for(i=0; i<row; i++)
     for(j=0; j<col; j++){
         dt = (*A).val(i,j);
          l = is;
       for(int m=0; m<n; m++){
         if(dt == legend_num[m]){
//            printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
           is++;
  }
     }
      if (l == is)
          printf("%g ", dt);
    }
    if (is != row*col)
       ler.terminate("Please check if lake rules cover all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
}

//********************** end of lake.h *********************************//

//high_con.h

class high_con{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
```

```
    bfmatrix *B;
    friend class target;
    high_con(char *ndname, char *rulef, network &, int r, int c);
    ~high_con();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   high_con::high_con(char *ndname, char *rulef, network &nw,
              int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("high_con rulebase set up successfully\n");
//   nw.write_netwk();
   int it = nw.terminalp("high_con");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("high_con");
   printf("form high_con up is to %s\n", p);
 }

rules *high_con::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
 }

void high_con::write_bfmap(){
    B->print();
  }

void high_con::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s\n then %s\n", f->pre, f->cln);
    printf("%g %g %g\n", f->bel, f->dis, f->unc);
    };
}

bfmatrix *high_con::apply_rules(char *upnd){
    rule *ru;
    int k, i, j;
    k = ruleb->count();
    bfn *belf = new bfn;
//    bfmatrix *temp = new bfmatrix(row, col);
    j = 0;
    for(i=0; i<k; i++){
    ru = (rule *)(*ruleb)[i];
    if((strcmp(upnd, ru->cln)) == 0){
         belf->bel = ru->bel;
         belf->dis = ru->dis;
         belf->unc = ru->unc;
      }
```

```
      else
        j++;
      }
      if(j > k){
        printf("fav_geol error: Sorry, I don't know %s\n", upnd);
        exit(1);
       }
       else
         return (pass_rule(B, belf));
//       return temp;
 }

high_con::~high_con(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//************************ end of high_con.h ********************//

//high_airmag.h

class high_airmag{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    high_airmag(char *ndname, char *rulef, network &, int r, int c);
    ~high_airmag();
    rules *rulb(char *);
//    bfmatrix *set_B(int,  int);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   high_airmag::high_airmag(char *ndname, char *rulef, network &nw,
              int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("high_airmag rulebase set up successfully\n");
//   nw.write_netwk();
   int it = nw.terminalp("high_airmag");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("high_airmag");
   printf("from high_airmag up is to %s\n", p);
 };

//bfmatrix *high_airmag::set_B(m, n){
//  bfmatrix *temp = new bfmatrix(m, n, 0);
//  return temp;
//}

rules *high_airmag::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
   return temp;
```

```
 }

void high_airmag::write_bfmap(){
    B->print();
  }

void high_airmag::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s\n then %s\n", f->pre, f->cln);
    printf("%g %g %g\n", f->bel, f->dis, f->unc);
    };
}

bfmatrix *high_airmag::apply_rules(char *upnd){
    rule *ru;
    int k, i, j;
    k = ruleb->count();
    bfn *belf = new bfn;
//    bfmatrix *temp = new bfmatrix(row, col);
    j = 0;
    for(i=0; i<k; i++){
    ru = (rule *)(*ruleb)[i];
    if((strcmp(upnd, ru->cln)) == 0){
         belf->bel = ru->bel;
         belf->dis = ru->dis;
         belf->unc = ru->unc;
      }
     else
       j++;
     }
     if(j > k){
       printf("fav_geol error: Sorry, I don't know %s\n", upnd);
       exit(1);
      }
      else
        return (pass_rule(B, belf));
//      return temp;
 }

high_airmag::~high_airmag(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//********************** end of high_airmag.h *********************//

//step_airmag.h

static error_handler ste("step_airmag error: ");

class step_airmag{
    char *mapname;
    int row, col;
    char *datfile;
    char *ansfile;
//    matrix *A;
```

```
//     bfmatrix *B;
      legend *lgnd;
      rules *ruleb;
      bfn prcn;
      double precision;
    public:
      step_airmag(char *, char *, char *, char *, network &, int, int);
      ~step_airmag();
      bfmatrix *apply_rules(char *);
      double get_precision(char *);
      legend *lgd(char *);
      rules *rulb(char *);
      matrix *dat(char *);
      void write_map(char *msg);
      void write_legend(char *msg);
      void write_rules(char *msg);
};

    step_airmag::step_airmag(char *map_name, char *datf, char *anscif,
                char *rulef, network &nw, int r, int c){
    mapname = malloc(strlen(map_name)+1);
    strcpy(mapname, map_name);
    row = r; col = c;
    datfile = malloc(strlen(datf)+1);
    strcpy(datfile, datf);
    ansfile = malloc(strlen(anscif)+1);
    strcpy(ansfile, anscif);
    lgnd = lgd(anscif);
    printf("step_airmag legend set up successfully\n");
    ruleb = rulb(rulef);
    printf("step_airmag rulebase set up successfully\n");
    nw.set_status("step_airmag", "step_airmag");
//    nw.write_netwk();
    int it = nw.terminalp("step_airmag");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("step_airmag");
    printf("form step_airmag up is to %s\n", p);
 }

matrix *step_airmag::dat(char *datfile){
    matrix *Inp = new matrix(row, col);
    Inp->readat(datfile);
    return Inp;
 }

legend *step_airmag::lgd(char *ansf){
    legend *temp = new legend;
    temp->set_leg(ansf);
    return temp;
 }

rules *step_airmag::rulb(char *rulf){
     rules *temp = new rules;
     temp->set_rules(rulf);
     return temp;
  }

double step_airmag::get_precision(char *ndnm){
    double temp;
```

```
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map and");
    printf("1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
  }
  }while(flag == 0);
 }
  prcn.bel = temp;
  prcn.dis = 0;
  prcn.unc = 1.0 - temp;
  return temp;
 }

//void step_airmag::write_map(char *msg){
//   A->print(msg);
// }

void step_airmag::write_legend(char *msg){
   leg_p *p;
   if(*msg) printf("%s\n", msg);
   int k = lgnd->count();
  for(int i=0; i<k; i++){
    p = (leg_p *)(*lgnd)[i];
     printf("%s  %g\n", p->leg_name, p->num);
  }
}

void step_airmag::write_rules(char *msg){
   rule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
               f->bel, f->dis, f->unc);
    }
}

step_airmag::~step_airmag(){
   delete mapname;
```

```
//    delete A;
//    delete B;
//    delete prcn;
    delete ruleb;
    delete lgnd;
}

bfmatrix *step_airmag::apply_rules(char *upndnm){
    double legend_num[40];
    double b[40], d[40];// u[40];
    int k, l, n, i, j, is;
    bfmatrix *B = new bfmatrix(row,col);
    rule *ru;
    leg_p *le;
    precision = get_precision("step_airmag");
    matrix *A = dat(datfile);
    printf("step_airmag data set up successfully\n");
    write_rules("charge rules");
    k = lgnd->count();
    l = ruleb->count();
    n = 0;
    for(i=0; i<l; i++){
      ru = (rule *)(*ruleb)[i];
      for(j=0; j<k; j++){
         le = (leg_p *)(*lgnd)[j];
         if(((strcmp(le->leg_name, ru->pre)) == 0) &&
            ((strcmp(upndnm, ru->cln)) == 0)){
           legend_num[n] = le->num;
           b[n] = ru->bel;
           d[n] = ru->dis;
//           u[n] = ru->unc;
           n++;
         };
       };
    };
    for (i=0; i<n; i++){
      printf("%g %g %g\n", legend_num[i],b[i],d[i]);
    }
    double dt;
    is = 0;
//    A->print("A");
    for(i=0; i<row; i++)
      for(j=0; j<col; j++){
          dt = (*A).val(i,j);
          l = is;
        for(int m=0; m<n; m++){
          if(dt == legend_num[m]){
//             printf("dt=%g ln=%g\n", dt, legend_num[m]);
            B->bfval(i,j,b[m],d[m]);
            is++;
  }
      }
       if (l == is)
          printf("%g ", dt);
        }
    if (is != row*col)
    ste.terminate("Please check if rules of step_airmag cover all legends!\n");
    delete A;
    return (pass_rule(B, &prcn));
}
```

```
//********************** end of step_airmag.h **********************//

//step_airmag.h

class image{
    int row, col;
    rules *ruleb;
    char *ndnm;
  public:
    bfmatrix *B;
    image(char *ndname, char *rulef, network &nw, int r, int c);
    ~image();
    rules *rulb(char *);
    bfmatrix *apply_rules(char *);
    void write_rules(char *msg);
    void write_bfmap();
};

   image::image(char *ndname, char *rulef, network &nw,
                int r, int c){
   ndnm = malloc(strlen(ndname));
   strcpy(ndnm, ndname);
   row = r; col = c;
   ruleb = rulb(rulef);
   printf("image rulebase set up successfully\n");
   int it = nw.terminalp("image");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("image");
   printf("from image up is to %s\n", p);
 }

bfmatrix *image::apply_rules(char *upnd){
     rule *ru;
     int k, i, j;
     k = ruleb->count();
     bfn *belf = new bfn;
//     bfmatrix *temp = new bfmatrix(row, col);
     j = 0;
     for(i=0; i<k; i++){
       ru = (rule *)(*ruleb)[i];
       if((strcmp(upnd, ru->cln)) == 0){
           belf->bel = ru->bel;
           belf->dis = ru->dis;
           belf->unc = ru->unc;
       }
       else
        j++;
       }
      if(j > k){
         printf("fav_geol error: Sorry, I don't know %s\\n", upnd);
         exit(1);
       }
       else
         return (pass_rule(B, belf));
//         return temp;
 }

rules *image::rulb(char *rulf){
   rules *temp = new rules;
   temp->set_rules(rulf);
```

```
   return temp;
 }

void image::write_bfmap(){
    B->print();
  }

void image::write_rules(char *msg){
   rule *f;
   if(*msg)
     printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (rule *)(*ruleb)[i];
    printf("if %s then %s (%g %g %g)\n", f->pre, f->cln,
                f->bel, f->dis, f->unc);
    }
}

image::~image(){
   delete ndnm;
//   delete B;
   delete ruleb;
}

//****************************** end of img.h ********************//

//img1.h

static error_handler img1_error("image1 error: ");

class image1{
    char *mapname;
    int row, col;
    char *datfile;
    char *nrulf;
    num_rul *ruleb;
    bfn prcn;
//    bfmatrix *B;
    double precision;
    void set_prcn();
  public:
    image1(char *, char *, char *, network &, int, int);
    ~image1();
    char *get_name(){return mapname;}
    bfmatrix *apply_rules(char *);
    matrix *dat(char *datf);
    num_rul *rles(char *);
    double get_precision(char *);
//    void write_map(char *msg);
    void write_rules(char *msg);
};

//image1::image1(){;}

   image1::image1(char *map_name, char *datf,
               char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
```

```
    strcpy(datfile, datf);
    nrulf = malloc(strlen(rulef)+1);
    strcpy(nrulf, rulef);
    ruleb = rles(rulef);
    printf("air_mag rulebase set up successfully\n");
//    nw.write_netwk();
    nw.set_status("image1", "image1");
    int it = nw.terminalp("image1");
    printf("terminalp=%d\n", it);
    char *p = nw.get_towhoup("image1");
    printf("from image1 up is to %s\n", p);
 }

num_rul *image1::rles(char *rulf){
    num_rul *rul = new num_rul;
    rul->set_nr(rulf);
    return rul;
 }

matrix *image1::dat(char *datf){
    matrix *Geo = new matrix(row, col);
    Geo->readat(datf);
    return Geo;
 }

double image1::get_precision(char *ndnm){
    double temp;
    int flag;
    int c;
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map");
    printf("and 1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
  }
  }while(flag == 0);
 }
  return temp;
 }

void image1::set_prcn(){
   prcn.bel = precision;
   prcn.dis = 0;
   prcn.unc = 1.0-precision;
```

```
    printf("precision is set to belief = %g\n",precision);
    printf("prcn = %g %g %g\n",prcn.bel,
                prcn.dis, prcn.unc);
}

//void image1::write_map(char *msg){
//    image1_map->print(msg);
// }

void image1::write_rules(char *msg){
    nrule *f;
    if(*msg)
        printf("%s\n", msg);
    int k = ruleb->count();
    for(int i=0; i<k; i++){
     f = (nrule *)(*ruleb)[i];
     printf("if %s %g %g\n then %s (%g %g %g)\n", f->pre,
            f->num1, f->num2, f->cln, f->bel, f->dis, f->unc);
     };
}

image1::~image1(){
    delete mapname;
//    delete B;
    delete ruleb;
//    delete image1_map;
//    delete prcn;
    delete datfile;
    delete nrulf;
}

bfmatrix *image1::apply_rules(char *upndnm){
    char *pre[40];
    double n1[40], n2[40], b[40], d[40];// u[40];
    int l, n, i, j, k, is;
    bfmatrix *B = new bfmatrix(row,col);
    nrule *ru;
    precision = get_precision("image1");
    set_prcn();
    printf("prcn = %g %g %g\n",prcn.bel,
                prcn.dis, prcn.unc);
    l = ruleb->count();
    n = 0;
    write_rules("image_1 rules");
    matrix *image1_map = dat(datfile);
//    write_map("image1_map");
//    for (i=0; i<10; i++){
//      printf("%10.7g", (*image1_map).val(10,i));
//    }
    printf("image1 data set up successfully\n");
//    write_rules("rules");
    printf("upnode is %s\n", upndnm);
    for(i=0; i<l; i++){
      ru = (nrule *)(*ruleb)[i];
          if ((strcmp(upndnm, ru->cln)) == 0){
           pre[n] = malloc(strlen(ru->pre)+1);
           strcpy(pre[n], ru->pre);
           n1[n] = ru->num1;
           n2[n] = ru->num2;
```

```
            b[n] = ru->bel;
            d[n] = ru->dis;
//            u[n] = ru->unc;
            n++;
          }
        }
    for (i=0; i<n; i++)
      printf("%s %g %g %g %g\n", pre[i],n1[i],n2[i],b[i],d[i]);
    is = 0;
    for (k=0; k<n; k++){
      printf("%g %g %g %g\n", n1[k],n2[k],b[k],d[k]);
      if ((strcmp(pre[k], "greater_than")) == 0){
       for(i=0; i<row; i++)
        for(j=0; j<col; j++){
          if ((*image1_map).val(i,j) >= n1[k]){
             B->bfval(i,j,b[k],d[k]);
             is++;
  }
      }
     }
      else if
         ((strcmp(pre[k], "less_than")) == 0){
       for(i=0; i<row; i++)
        for(j=0; j<col; j++){
          if ((*image1_map).val(i,j) <= n1[k]){
             B->bfval(i,j,b[k],d[k]);
             is++;
  }
      }
    }
     else if
        ((strcmp(pre[k], "between")) == 0){
       for(i=0; i<row; i++)
        for(j=0; j<col; j++){
          if (((*image1_map).val(i,j) >= n1[k]) &&
              ((*image1_map).val(i,j) <= n2[k])){
             B->bfval(i,j,b[k],d[k]);
             is++;
  }
      }
    }
    else
      img1_error.terminate("Sorry, I don't know %s.\n", pre[k]);
   }
   if (is < row*col)
    img1_error.terminate("Please check if rules cover all pixel values!\n");
    for (i=0; i<n-1; i++)
      delete pre[i];
      delete n1;
      delete b;
      delete d;
      delete n2;
      delete image1_map;
    return (pass_rule(B, &prcn));
}

//********************** end of img1.h *******************************//

//img2.h

static error_handler img2_error("image2ic error: ");
```

```
class image2{
    char *mapname;
    int row, col;
    char *datfile;
    char *nrulf;
//    matrix *image2_map;
    num_rul *ruleb;
    bfn prcn;
//    bfmatrix *B;
    double precision;
    void set_prcn();
  public:
   image2(char *, char *, char *, network &, int, int);
    ~image2();
    char *get_name(){return mapname;}
    bfmatrix *apply_rules(char *);
    matrix *dat(char *datf);
    num_rul *rles(char *);
    double get_precision(char *);
//    void write_map(char *msg);
    void write_rules(char *msg);
};

//image2::image2(){;}

   image2::image2(char *map_name, char *datf,
                  char *rulef, network &nw, int r, int c){
   mapname = malloc(strlen(map_name)+1);
   strcpy(mapname, map_name);
   row = r; col = c;
   datfile = malloc(strlen(datf)+1);
   strcpy(datfile, datf);
   nrulf = malloc(strlen(rulef)+1);
   strcpy(nrulf, rulef);
   ruleb = rles(rulef);
   printf("image2 rulebase set up successfully\n");
//   nw.write_netwk();
   nw.set_status("image2", "image2");
   int it = nw.terminalp("image2");
   printf("terminalp=%d\n", it);
   char *p = nw.get_towhoup("image2");
   printf("from image2 up is to %s\n", p);
 }

num_rul *image2::rles(char *rulf){
   num_rul *rul = new num_rul;
   rul->set_nr(rulf);
   return rul;
 }

matrix *image2::dat(char *datf){
   matrix *Geo = new matrix(row, col);
   Geo->readat(datf);
   return Geo;
 }

double image2::get_precision(char *ndnm){
   double temp;
   int flag;
   int c;
```

```
    flag = 0;
    while(flag == 0){
    printf("Please enter a number between 0 to 1 which shows");
    printf("the certainty based on the accuracy of the %s map,",ndnm);
    printf("0 means nothing believeable on this map");
    printf("and 1 means you totally believe the map is true.\n");
    do {
    scanf("%lf", &temp);
    if(temp >1)
      printf("The number you given greater than 1.\nPlease reenter:");
    else if(temp < 0)
       printf("The number you entered less than 0.\nPlease reenter:");
    else{
     printf("Your certainty about %s map is %g\n", ndnm, temp);
     c = getchar();
     if(c == 'y')
        flag = 1;
     else
       printf("Is this number looks ok for you?(y/n)");
       printf("If not, please enter your new number:");
  }
  }while(flag == 0);
 }
  return temp;
 }

void image2::set_prcn(){
   prcn.bel = precision;
   prcn.dis = 0;
   prcn.unc = 1.0-precision;
   printf("precision is set to belief = %g\n",precision);
   printf("prcn = %g %g %g\n",prcn.bel,
                prcn.dis, prcn.unc);
}

//void image2::write_map(char *msg){
//    image2_map->print(msg);
// }

void image2::write_rules(char *msg){
   nrule *f;
   if(*msg)
       printf("%s\n", msg);
   int k = ruleb->count();
   for(int i=0; i<k; i++){
    f = (nrule *)(*ruleb)[i];
    printf("if %s %g %g\n then %s (%g %g %g)\n", f->pre,
           f->num1, f->num2, f->cln, f->bel, f->dis, f->unc);
    };
}

image2::~image2(){
   delete mapname;
//    delete B;
   delete ruleb;
//    delete image2_map;
//    delete prcn;
   delete datfile;
   delete nrulf;
```

```
}

bfmatrix *image2::apply_rules(char *upndnm){
   char *pre[40];
   double n1[40], n2[40], b[40], d[40];// u[40];
   int l, n, i, j, k, is;
   bfmatrix *B = new bfmatrix(row,col);
   nrule *ru;
   precision = get_precision("image2");
   set_prcn();
   printf("prcn = %g %g %g\n",prcn.bel,
                prcn.dis, prcn.unc);
   l = ruleb->count();
   n = 0;
   write_rules("image2 rules");
   matrix *image2_map = dat(datfile);
//   write_map("image2_map");
//   for (i=0; i<10; i++){
//     printf("%10.7g", (*image2_map).val(10,i));
//   }
   printf("image2 data set up successfully\n");
//   write_rules("rules");
   printf("upnode is %s\n", upndnm);
   for(i=0; i<l; i++){
     ru = (nrule *)(*ruleb)[i];
         if ((strcmp(upndnm, ru->cln)) == 0){
          pre[n] = malloc(strlen(ru->pre)+1);
          strcpy(pre[n], ru->pre);
          n1[n] = ru->num1;
          n2[n] = ru->num2;
          b[n] = ru->bel;
          d[n] = ru->dis;
//          u[n] = ru->unc;
          n++;
        }
      }
   for (i=0; i<n; i++)
     printf("%s %g %g %g %g\n", pre[i],n1[i],n2[i],b[i],d[i]);
   is = 0;
   for (k=0; k<n; k++){
     printf("%g %g %g %g\n", n1[k],n2[k],b[k],d[k]);
     if ((strcmp(pre[k], "greater_than")) == 0){
      for(i=0; i<row; i++)
       for(j=0; j<col; j++){
         if ((*image2_map).val(i,j) >= n1[k]){
            B->bfval(i,j,b[k],d[k]);
            is++;
  }
       }
     }
     else if
        ((strcmp(pre[k], "less_than")) == 0){
      for(i=0; i<row; i++)
       for(j=0; j<col; j++){
         if ((*image2_map).val(i,j) <= n1[k]){
            B->bfval(i,j,b[k],d[k]);
            is++;
  }
       }
    }
```

```
ERROR: invalidrestore
OFFENDING COMMAND: restore], "between")) == 0){
        for(i=0; i<row; i++)
STACK:   for(j=0; j<col; j++){
           if (((*image2_map).val(i,j) >= n1[k]) &&
-savelevel-    ((*image2_map).val(i,j) <= n2[k])){
((pass_rule)
              B->bfval(i,j,b[k],d[k]);
              is++;
    }
        }
      }
      else
        img2_error.terminate("Sorry, I don't know %s.\n", pre[k]);
     }
     if (is < row*col)
      img2_error.terminate("Please check if rules cover all image2 values!\n");
      for (i=0; i<n-1; i++)
        delete pre[i];
        delete n1;
        delete b;
        delete d;
        delete n2;
        delete image2_map;
      return (B, &prcn));
   }

  //**************** end of img2.h ***************************************//
```