# USE OF THE FIXED POINT THEOREM TO MINE DATA FROM A KNOWLEDGE-BASE

By

LIWEI CAI

Bachelor of Science

Beijing Institute of Chemical Technology

Beijing, China

1992

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
In Partial Fulfillment of
The Requirements of
The Degree of
MASTER OF SCIENCE
May, 1998

OKLAHOMA STATE UNIVERSITY

# USE OF THE FIXED POINT THEOREM TO MINE DATA FROM A KNOWLEDGE-BASE

Thesis Approved:

[illegible]

Thesis Adviser

H. Lu

Jacques LaFrance

Wayne B. Powell

Dean of Graduate College

# PREFACE

"One gram of knowledge is worth of tons of gold". The desire to obtain knowledge from collected data has generated a need for new techniques and tools that can transform the processed data into useful information and knowledge intelligently and automatically. Consequently, data mining has become a research area with increasing importance. Data mining, which is also known as knowledge discovery in databases, is the process of extracting and refining implicit, previously unknown and potentially useful knowledge from databases. Much research has set mining information and knowledge from databases as a key research topic. It also becomes an important area with major revenues opportunity for companies. Many researchers in the area of database systems, data warehousing, statistics, artificial intelligence, and expert systems have shown their interest in data mining. In response to such a demand a new data mining technique, the use of the perfect fixed point theorem is discussed and validated in this thesis. By using the perfect fixed point theorem in knowledge-bases, people can obtain some useful (previously unknown) information. It can help decision makers to make more accurate predications. Also, it can save time and money for decision makers and the researchers in other scientific and technology fields.

## ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. G. E. Hedrick, my major adviser, for his constant inspiration and encouragement throughout my graduate study. None of this would have been possible without his consistent advice and generous aid. I also wish to express my sincere gratitude to Dr. H. Lu for her guidance and support during my graduate study. Thanks to Dr. J. Lafrance for serving on my committee.

I am grateful for the help I have received from many individuals. In particular, I wish to extend my gratitude to Mr. Yuwen Lin and Mr. Nanda Muhammad for their kind help.

My parents, Guangdao Cai and Baozhen Wang, deserve my deep gratitude for being a constant source of support in my life.

Last, but not least, my appreciation goes to my husband Wuzi Gao for his constant love, encouragement, and support.

TABLE OF CONTENTS

LIST OF TABLES

# CHAPTER I

# INTRODUCTION

## 1.1 Promising Data Mining Field

With the development of modern society, information and data has become more important than ever. Millions of databases have been used in business management, government administration, scientific and engineering data management, as well as many other applications. This explosive growth in data and databases has generated an urgent need for new techniques and tools that can transform the processed data into useful information and knowledge intelligently and automatically. Consequently, data mining has become a research area with increasing importance [CHY97].

Data mining, which is also known as a stage of knowledge discovery in databases, is the process of extracting and refining implicit, previously unknown, and potentially useful knowledge from a large database [FSPSmU96, SPFr91]. Mining information and knowledge from a large database then investigating from different angles, allows the database to serves as a rich and reliable source for knowledge generation and verification. Therefore, many projects have set mining information and knowledge from large databases as their key research topic. It's also becoming an important area with major revenue opportunities for companies. Researchers in different areas, such as database systems, data warehousing, statistics, on-line analysis processing, artificial intelligence, expert systems, and data visualization have shown their interest in data mining [AdZa96].

## 1.2 Theme of the Thesis

In response to the demand of developing a new data mining technique, a new method, use of the fixed point theorem, is discussed and validated in this thesis. The fixed point theorem includes *naïve, semi-naïve, perfect fixed point* and other fixed point theorems developed by the researchers in the University of Wisconsin at Madison [RstSu94]. Because the *perfect fixed point* algorithm is the most general and robust, its idea is used in this work. Mining databases to develop a model can be performed in either a top-down or a bottom-up fashion. The bottom-up fashion will be used here, because using bottom-up evaluation of datalog (defined in 2.3.1) always terminates and computes the answer to the query in linear time complexity [RaStSu94]. Through the use of the perfect fixed point theorem as an example, it can be seen when the perfect fixed point theorem is used in data mining, it is a easy to use, effective, and robust tool.

## 1.3 Thesis Organization

This thesis is organized in the following way: chapter II provides the literature reviews of the basic concepts that appear in the thesis; what other people did in data mining, datalog, and an overview of fixed point theorem. Chapter III describes, in detail, the approach and procedure by illustrating an example of mining useful information from a knowledge base using the perfect fixed point theorem. Chapter IV shows the result of the modified perfect fixed point algorithm, then discuss it by comparing to unmodified perfect fixed point algorithm. Chapter V summarizes the significance of this new data mining method and describe the possible future works.

# CHAPTER II

# LITERATURE REVIEW AND BASIC CONCEPTS

## 2.1 BRIEF REVIEW

In the past decade, with the development of networks and the client/server revolution, people became able to collect and generate data more easily than before. Millions of databases are created in business management, government administration, scientific and engineering fields, and many other applications. Such volumes of data clearly overwhelm the traditional manual methods of data analysis such as spreadsheets and ad-hoc queries. Those older methods can create informative reports from data, but cannot analyze the contents of those reports to focus on important information (or knowledge). In addition, it requires substantial time from one or more people to obtain the useful information from the reports [AdZa96]. Consequently, there is an urgent need for new techniques and tools that can transform the processed data into useful information and knowledge intelligently and automatically. As a result, the concept of knowledge discovery in database (KDD), or data mining, was developed as one of the most useful such techniques.

The interest in KDD, or data mining, has been increasing in recent years. The trend can be observed from the frequently held workshops, international conferences on knowledge discovery and data mining, new created magazines such as *DW for Data Warehousing Management*, *Knowledge Discovering in Database*, and increasing numbers of published papers in this area.

Currently, there are many popular data mining methods including query tools, decision trees, neural networks, and so on. Although they are different when they are applied to real life applications, they still have something in common. All of them start from the databases, then use the query tool (such as SQL) to get some basic rules to add to the database (thus, a knowledge-base is generated), then more useful information can be extracted from the knowledge-base.

Based on this idea, a new data mining method, use of the perfect fixed point theorem is validated in the thesis by both the illustration of a common example adapted from the data mining literature and a real life example.

## 2.2 Basic Concepts

There are three key words in the title of the thesis: knowledge-base, data mining, and the fixed point theorem. These three key concepts must be explained.

### 2.2.1 Knowledge-base

Currently, there is a growing interest in the design and implementation of knowledge-base systems. Unfortunately, at present, there is no widely accepted definition of the term knowledge-base, so this term has been used by different people to mean different things. The meaning of "a collection of simple facts and general rules representing some universe of discourse" is used for a knowledge-base [Frost86]. It may consist of more facts than general rules, such as a knowledge-base representing a train timetable. On the other hand, it may consist of more general rules than simple facts, such as, a knowledge-base representing good chess moves. The rules in a knowledge-base can be implemented using the rule-based language, Prolog.

Most knowledge-bases are distinct from conventional databases in that they typically consist of explicitly stated general rules as well as explicitly stated simple facts. Databases, on the other hand, typically consist of a large number of explicitly stated simple facts together with a relatively small number of implicitly stored general rules. When the common knowledge to be integrated into the database is expressed as rules, the relational database becomes a knowledge-base [FPSSmU96]. The tuples stored in the database relations constitute the extensional database (EDB). The rules, which are deduced from the relations, are defined as the intensional database (IDB).
Managing knowledge-base is not easy. Some complex research problems must be solved. These problems are usually at the crossroads of artificial intelligence and database theory. Georges Gardarin described the approaches to solve these problems from the theoretic and the practical views in his book *Relational Database and Knowledge bases* [Gard89] as follows:

> From a theoretical point of view, the approaches using logic entail a good understanding of data and rule models. From a practical point of view, the approaches with prototypes should yield important developments that can be used to experiment with knowledge-base management systems (KBMSs) that support not only relations and rules but also complex objects.

### 2.2.2 Data-mining

Usually, the knowledge discovery process consists of six stages: (1) Data selection, (2) Cleaning, (3) Enrichment, (4) Coding, (5) Data mining, and (6) Reporting. The fifth stage, data mining is the phase of real discovery [AdZa96].

Data mining is the process of extracting and refining useful knowledge from large databases. The extracted information can be used to form a prediction or classification model, identify trends and associations, refine an existing model, or provide a summary of the database(s) being mined. A number of data mining techniques, e.g., rule induction, neural networks, and conceptual clustering [Salton71], have been developed and used individually in domains ranging from space data analysis to financial analysis. Data Mining also refers to the application of algorithms for extracting patterns from data without additional steps of the KDD (knowledge discovery of database) process (such as incorporating appropriate prior knowledge and proper interpretation of the results).

Making effective and accurate decisions directly from data stored in database is difficult for two reasons. "First, these databases are frequently very large. Second, the database is usually described in terms of low-level concepts and relations that are cognitively distant from the concepts used by the decision maker" [AdZa96].

Mining databases to develop a model can be performed in either a top-down or a bottom-up fashion. During top-down data mining the analyst defines concepts and rules in terms of the entities and relations included in the database, then proceeds by progressively extending these concepts to higher-level ones. During bottom-up data mining the system automatically creates concepts and rules that either specialize or generalize those defined by the analyst. Therefore, the data mining process is performed with three operations: (1) generation, and testing of concepts and rules hypothesized by the analyst; (2) automatic, or semi-automatic, generation and refinement of concepts and rules by the data mining system; and (3) integration of the best of both the manually generated and the automatically discovered concepts and rules into a cohesive model [FshSmU96].

Data mining has two "high-level" primary goals, prediction and description. Prediction involves using one or more variables or fields in the database to predict unknown or future values of other variables of interest.

The goals of prediction and description are achieved by using the following primary data mining tasks.

**Classification** is the learning a function that maps (classifies) a data item into one of several predefined classes.

**Regression** is the learning a function which maps a data item to real-valued prediction variable.

**Clustering** is a common descriptive task that one seeks to identify a finite set of categories or clusters to describe the data.

**Summarization** involves methods for finding a compact description of a subset of data.

Algorithms need to be constructed to fulfill the task of data mining. There are three components in any data mining algorithm: model representation, model evaluation, and searching.

## 2.2.3 Fixed Point theorem

The computation of the least fixed point of a transformation has numerous applications in computer science, including formal languages, databases, and artificial intelligence. The basic definition of a *fixed point theorem* is: if f is a map is from X to X, then for some point P of X, we have f (P)=P. Such a point P is called a fixed point of f.

In the evaluation of database or knowledge-base, a simple way to compute a relation which is defined by a recursive rule of the form r = f (r), where f is a relational algebra expression, consists of executing the following program:

```
r = $\phi$;
while "r changes"
do r := f (r) [BrMy86].
```

This kind of program is usually called *relational algebra program* (RAP). The above program is called the *naïve evaluation* of the fixed point of f. A more effective evaluation algorithm to get the least fixed point is called *Semi-Naïve Evaluation* or *bottom-up evaluation*. According to some researchers, the bottom-up evaluation is more effective than top-down evaluation. The main advantage is (1) bottom-up evaluation makes no more inferences than top-down for range-restricted programs (see Appendix A); (2) for a restricted class of programs, which properly includes safe datalog (defined in 2.3.1), the cost of bottom-up evaluation is never worse than a constant times the cost of a corresponding top-down evaluation (and can be much better than prolog for many programs) [RaSu91].

## 2.3 Terms

There are several terms used with the fixed point theorem. They are explained as follows:

### 2.3.1 Datalog

The term, "datalog", was coined to suggest a version of Prolog suitable for database systems. According to Ullman [Ullm88], it differs from Prolog in several respects:

(1) Datalog does not allow function symbols in arguments. Rather, datalog allows only variables and constants as arguments of predicates.
(2) The "meaning" of datalog programs follows the model-theoretic point of view. Prolog, however, has a computational "meaning", which, as we discussed, can deviate in some cases from either the model-theoretic or proof-theoretic meanings.

The underlying mathematical model of data for datalog is essentially the same as that of the relational model. Predicate symbols in datalog denote relations. These relations do not have attributes with which to name their columns. They are in the set-of-lists sense, where components appear in a fixed order, and reference to a column is only by its position among the arguments of a given predicate symbol.

### 2.3.2 Extensional and Intentional Predicates

Another distinction between the relational and datalog model is that in datalog, there are two ways relations can be defined. A predicate whose relation is stored in the database is called an extensional database (EDB) relation, while one defined by logical rules is called an intentional database (IDB) relation. IDB relations are usually temporary relations that hold intermediate results computed during the execution of a query. We assume that each predicate symbol either denotes an EDB relation or an IDB relation, but not both. In the relational model, all relations are EDB relations.

Given a set of relations for the EDB predicates, say $R_1,\ldots,R_k$, a fixed point of the datalog equations (with respect to $R_1,\ldots,R_k$), is a solution for the relations corresponding to the IDB predicates of those equations. According to Ullman [Ullm88], datalog programs each have a unique minimal model containing any given EDB relations, and this minimal model is also the unique minimal fixed point, with respect to those EDB relations, of the corresponding equations.

# CHAPTER III

# APPROACH AND PROCEDURE

## 3.1 Algorithm

The idea of the Perfect Fixed Point algorithm is used for data mining in this thesis. This algorithm evaluates relations for safe (see Appendix A), stratified datalog programs. Rules are stratified, if whenever there is a rule with head predicate p and a negated subgoal with predicate q, there is no path in the dependency graph from p to q. The input for it is a datalog program whose rules are safe, rectified (see Appendix A), and stratified and relations for all the EDB predicates of the program. Its outputs are relations for all the IDB predicates forming a minimal fixed point of the datalog program. This algorithm is discussed in some books and publications by Ullman, Ramakrishman et al, and Piatetsky-Shapiro et al [Ullm88, RaSu91, PSFr91]. The perfect algorithm used in this thesis is almost the same, except with the author's modification for the current task.

The method used is: first, compute the stratification for the datalog program by the stratification algorithm and divide all the rules in the datalog program in groups according to their stratum. The stratification algorithm from Ullman's book[Ullm88] is as follows:

```
for each predicate p do
        stratum[p] := 1;
    repeat
        for each rule r with head predicate p do begin
            for each negated subgoal of r with predicate q do
                stratum[p] := max(stratum[p], 1+stratum[q]);
            for each nonnegated subgoal of r with predicate q do
                stratum[p] := max(stratum[p], stratum[q]);
        end
    until there are no changes to any stratum
        or some stratum exceeds the number of predicates
```

If a rule with negated subgoal is not stratified, then the logic program will enter an infinite loop. Since not every logic program with negations is stratified, it is useful to have an algorithm to test for stratification.

Second, for each stratum I, in turn, do the listed below steps. When stratum I is reached, the relations for the IDB predicates at lower strata have been already computed, and the relations for the EDB predicates are given. (Actually, they are acquired from the result of querying to the initial relational database.) Thus, in particular, if a rule at stratum I has a negated subgoal, the relation for that subgoal is known. The perfect fixed point algorithm includes the following four steps:

(1) Check all the rules with the same strata one by one.

(2) Consider each nonnegated subgoal in a rule for stratum I. If that subgoal is an EDB predicate or an IDB predicate at a stratum below I or IDB predicate at the same stratum but already known, use the relation already known for that predicate.

(3) For each negated subgoal in a rule for stratum I, the IDB relation must have been computed, because the stratum of the predicate for the negated subgoal is less than I. (At the moment the head predicate of the negation subgoal is to be

calculated, all the IDB relations for the negated subgoal are actually already calculated). After a check is performed, if the IDB relations of the negated subgoal are ready, they can be used.

(4) Use the Semi-Naïve algorithm to compute the relations for the IDB predicates of stratum I, treating all the subgoals whose relation were obtained in either step (2) or step(3), as if they were EDB relations with the values given by those steps.

### 3.1.1 Semi-naïve Algorithm

There are two effective ways to evaluate the rules in the datalog program with recursive rules: the naïve algorithm and the semi-naïve algorithm. The naïve algorithm starts by assuming all the IDB relations are empty. When the evaluation of the rules is applied to the current IDB relations based on the EDB relations, some new IDB relations can be obtained. This process is repeated until at some point (the fixed point), the IDB is never changed. The algorithm by Ullman [Ullm88] is as follows:

(1) Input: A collection of datalog rules with EDB predicates $r_1, \ldots r_k$ and IDB predicates $p_1, \ldots, p_m$. Also, a list of relations $R_1, \ldots R_k$ to serve as values of the EDB predicates.

(2) Output: The least fixed point solution to the datalog equations obtained from these rules.

(3) Method:

```
**for i** := 1 to m **do**
        $P_i := \phi$;
**repeat**
        **for** i := 1 to m **do**
                $Q_i := P_i$;
        **for** i := 1 to m **do**
                $P_i := EVAL(p_i, R_1, \ldots, R_k, Q_1, \ldots, Q_m)$;
**until** $P_i = Q_i$ for all i, $1 \le i \le m$;
```

Output $P_i$'s;

The naïve algorithm is very useful in the evaluation of a datalog program. It's not efficient as we would desire because when it performs the evaluation, it always obtains some repeated results. There's a more useful algorithm than naïve algorithm is called semi-naïve algorithm [Ullm88]. The semi-naïve algorithm takes advantage of incremental relations. It evaluates the rules in increments of one by one. When there are no more new IDB relations generated, then the evaluation reaches a fixed point and the evaluation stops. This fixed point is also the least fixed point for the datalog program. According to Ullman [Ullm88], its steps are as follows:

(1) Input: A collection of datalog rules with EDB predicates $r_1, \ldots r_k$ and IDB predicates $p_1, \ldots, p_m$. Also, a list of relations $R_1, \ldots R_k$ to serve as values of the EDB predicates.

(2) Output: The least fixed point solution to the datalog equations obtained from these rules.

(3) Method:

```
for i := 1 to m do begin
    $\Delta P_i := EVAL(p_i, R_1, \ldots, R_k, \phi, \ldots \phi)$;
    $P_i := \Delta P_i$;
End

repeat
    for i := 1 to m do
        $\Delta Q_i := \Delta P_i$;
    for i := 1 to m do begin
        $\Delta P_i := EVAL(p_i, R_1, \ldots, R_k, \Delta Q_1, \ldots, \Delta Q_m)$;
        $\Delta Q_i := \Delta P_i - P_i$;
    end
    for i := 1 to m do
        $\Delta Q_i := P_i \cup \Delta P_i$;
until $\Delta P_i = \phi$ for all I;

Output $P_i$'s;
```

## 3.2 Languages

### 3.2.1 Mini SQL

Test programs are created to illustrate how to mine data from the knowledge base. The EDB for the knowledge bases are facts from a initial relational database. The relational database can be created by any kind of language.

SQL (Structured Query Language) is used for testing the code. The original version of SQL was developed at IBM's San Jose Research Laboratory (now the Almaden Research Center). This language, originally called Sequel, was implemented as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL [KoSi91]. SQL has clearly established itself as the standard relational database language. SQL provides commands for a variety of task including:

(1) Querying data

(2) Inserting, updating and deleting rows in a table

(3) Creating, replacing, altering and dropping objects

(4) Controlling access to the database and its objects

(5) Guaranteeing database consistency and integrity

Mini SQL is a light weight database engine developed by David J. Hughes at Bond University, Australia. It has been designed to provide fast access stored data with low memory requirements. As its name implies, Mini SQL offers a subset of SQL as its query interface. Although it only supports a subset of SQL, everything it supports is in accordance with the ANSI SQL specification [Hugh97].

Mini SQL has a C API (see Appendix A). This API allows any C program to communicate with the database engine. The API functions are accessed by including the msql.h header file into the C program and by linking against the Mini SQL library [10]. The main API function used in this thesis is as follows:

**int msqlConnect(char *host)**

msqlConnect forms an interconnection with the Mini SQL engine. Its one argument is the host name of Mini SQL server (a.cs.okstate.edu). If the Mini SQL engine is one of the localhost, then NULL should be used as the parameter.

**int msqlSelectDB(int sock, char *dbName)**

msqlSelectDB instructs the Mini SQL engine which database is used for processing. Sock is the return value of msqlConnect. Since the function returns a unique socket descriptor, one client application can access multiple databases at one time.

**int msqlQuery(int sock, char *query)**

Queries are sent to the engine over the connections associated with sock as plain text strings using msqlQuery. The query results are stored in the API buffer for another function to retrieve later.

**m_result *msqlStoreResult()**

Data returned by a SELECT query are stored in the API buffer. msqlStoreResult retrieves the data from the buffer and puts it into an m_result structure as follows:

```
typedef struct result_s {
        m_data  *queryData,
          *cursor;
m_fdata *fieldData,
          *fieldCursor;
int       numRows,
          numFields;
} m_result;
```

In the structure m_result, *queryData will be used for retrieving the data which are selected row by row. "filedData" points to a structure called m_data. The structure of m_data is as follows:

```
typedef struct  m_data_s {
          int            width;
          m_row          data;
          struct m_data_s *next;
} m_data;
```

The retrieved data are stored temporally in the "data" of m_data and then are processed by a C program to have the form of "predicate (data, data…)". At this time, the query results are transferred into EDB rules for the knowledge-base.

## 3.2.2 BinProlog:

To perform the evaluation of logic rules, a kind of logic language is used. Prolog is chosen for the evaluations presented in this thesis. Prolog is a practical and efficient implementation of many aspects of logic program execution. A Prolog program consists of a set of clauses, where each clause is either a fact about the given information or a rule about how the solution may relate to or is inferred from the given facts.

BinProlog was developed by Paul Tarau at the University of Moncton, Canada, and is based on his BinWAM abstract machine, a specialization of the WAM for the efficient execution of binary programs [Tarau97].

BinProlog is a fast and compact Prolog compiler, based on the transformation of Prolog to binary clauses. BinProlog5.75 is probably the first Prolog system to feature dynamic recompilation. BinProlog is a fairly robust and complete Prolog implementation featuring both C-emulated execution and generation of standalone applications by compilation into C.

With BinProlog5.75, it is possible to compile user applications separately and link them with the emulator library and the "C-ified" (see Appendix A) compiler. This allows creation of a fully C-ified application in a few seconds.

To compile the BinProlog program into C, a makefile is created. This makefile is a template to create a C program from an original BinProlog program and generate an executable file at the same time. When users want to compile their BinProlog programs into C, they type **make PROJ=filename** (without the postfix ".pro"). The correspondencing ".c", ".h", ".o" and ".exe" files are created. To combine the BinProlog programs with C, people can call the BinProlog program in C by using "**system (exe file)**" in the C program. During its execution, the C program can jump into the BinProlog program automatically. After the execution of the BinProlog program, it will jumps back to execute other parts of the program.

## 3.3 Examples and Program Implementation

In order to show how to use fixed point theorem to mine data from a knowledge base, the following example adapted from many publications is used. In this thesis, it is called the "family relation knowledge-base". It shows how to find other relations, such as siblings, cousins , people who are related and the people who are related but not cousins based solely on the parents relations. It consists of the facts (EDB relations) and the rules (IDB relations). The EDB relations can be acquired by sending queries to the relational database. The results of the queries are the facts of the knowledge base. The rules are designed and bounded according what kind of hidden and unknown information the "miner" wants to know.The EDB relations for this knowledge base are parents. They are in an initial relational database that was created by Mini SQL. Suppose that there is table called par-chd (table1) in the database including information about parents and children. The relation of parents can be obtained by sending the query,

**SELECT child, parent FROM parent**

to the database. The results are stored in a structure called m_result (show in Chapter 3). Then through C program, these results can be stored into a file as EDB relations:

| child | Parent |
|---|---|
| alice | Mary |
| john | Mary |
| tom | Alice |
| ann | John |
| judy | Tom |
| doug | Ann |

Table 1. par-chd

```
parent(alice,mary).
parent(ann,john).
parent(ann,john).
parent(john,mary).
parent(judy,tom).
parent(tom,alice).
```

Together with the following IDB rules, the knowledge base is created:

```
sibling(X,Y) :- parent(X,Z), parent(Y,Z), X\=Y.
cousin(X,Y) :- parent(X,A), parent(Y,B), sibling(A,B).
cousin(X,Y) :- parent(X,A), parent(Y,B), cousin(A,B).
related(X,Y) :- sibling(X,Y).
related(X,Y) :- parent(X,C), related(Y,C).
related(X,Y) :- parent(Y,D), related(D,X).
jrel(X,Y) :- related(X,Y), not(cousin(X,Y)).
```

By the use of the perfect fixed point algorithm described in Chapter III, more data can be mined from the knowledge base. The rules at the same stratum are put into a temporary BinProlog file at the same time. The evaluation of the rules whose subgoals have been satisfied already builds a BinProlog file. The BinProlog file will be compiled by the "makefile". A C file and an executable file are generated at the same time. The executable file is called from the C program to run the BinProlog program. The program stops at a point (the least fixed point) when there are no more new facts to be added to the knowledge base.

For the purpose of showing how the perfect fixed point algorithm is used, a real life application follows. This example is called the "potential customer knowledge-base". It finds the potential customers for a new product based on the current customers of the same company. Assume there are two tables containing the information about the current customers' information (table 2) and the information about their friends (table 3). The potential customers are supposed to be the current customers who are females and the customer for category a and the female friends of the potential customers. Also, the serious buyers are defined to be the potential customers who are not living in district four. The EDB rules of the knowledge base can be obtained by sending queries as follows to the database:

```
SELECT cname, catagory FROM c_customer
SELECT cname FROM c_customer WHERE gender='f'
SELECT fname FROM c_friend WHERE gender='f'
SELECT cname, fname FROM c_friend
SELECT cname, district FROM c_customer
SELECT fname, district FROM c_friend
```

The IDB rules in this knowledge base are:

```
cust(X):-ccust(X,Y), Y=a.
pcust(X) :- cust(X), female(X).
pcust(X) :- friend(Y,X), female(X), pcust(Y).
sbuyer(X) :- pcust(X), not(live(X,4)).
```

Together with the EDB relations (shown in Appendix B), the needed knowledge base is formed. As in the last example, apply the perfect fixed point algorithm to this knowledge base until there is no more new fact added (reaches a least fixed point).

| cid | cname | gender | district | category |
|---|---|---|---|---|
| 1 | anna_abbott | f | 1 | a |
| 2 | jane_bain | f | 1 | a |
| 3 | brain_boling | m | 2 | b |
| 4 | john_clawson | m | 3 | a |
| 5 | amy_foutch | f | 4 | a |
| 6 | jeff_ford | m | 2 | b |
| 7 | brain_kerler | m | 1 | a |
| 8 | steve_laner | m | 2 | b |
| 9 | cindy_nelson | f | 3 | a |
| 10 | dustin_patrick | m | 4 | a |
| 11 | laura_clinton | f | 3 | b |
| 12 | jenny_baker | f | 4 | a |
| 13 | john_chain | m | 2 | b |
| 14 | ryan_west | m | 4 | b |
| 15 | cindy_stone | f | 3 | a |
| 16 | polly_johnson | f | 1 | a |
| 17 | greg_trane | m | 2 | a |
| 18 | linda_wu | f | 1 | a |
| 19 | mary_wagner | f | 4 | b |
| 20 | judy_tree | f | 1 | a |

Table 2 C_customers

| fid | cname | fname | gender | district |
|---|---|---|---|---|
| 1 | anna_abbott | chris_lando | m | 1 |
| 2 | jane_bain | jenny_barker | f | 2 |
| 3 | brain_boling | fred_grason | m | 3 |
| 4 | john_clawson | shiley_hook | f | 1 |
| 5 | amy_foutch | linda_shaw | f | 1 |
| 6 | jeff_ford | john_tylar | m | 2 |
| 7 | brain_kerler | alice_johnson | f | 3 |
| 8 | steve_laner | doug_neily | m | 1 |
| 9 | cindy_nelson | john_dean | m | 3 |
| 10 | dustin_patrick | sharron_Lou | f | 2 |
| 11 | laura_clinton | amy_foutch | f | 3 |
| 12 | jenny_baker | diana_bain | f | 4 |
| 13 | john_chain | greg_laner | m | 2 |
| 14 | ryan_west | kim_stone | f | 4 |
| 15 | cindy_stone | linda_smith | f | 4 |
| 16 | polly_johnson | chris_lin | m | 1 |
| 17 | greg_trane | alice_tree | f | 2 |
| 18 | linda_wu | mary_wayn | f | 1 |
| 19 | mary_wagner | tina_shaw | f | 3 |
| 20 | judy_tree | tom_smith | m | 1 |

Table 3. C_friend

# CHAPTER IV

# RESULTS AND DISCUSSION

## 4.1 Results

After the execution of the program, the new facts added to the knowledge base shown in appendices C and D.

## 4.2 Discussions

### 4.2.1 Data Mining Versus Query Tools

Query tools (SQL) and data mining tools are complementary. A data mining tool does not replace a query tool, but it does give the user many additional possibilities. For a large database, if one wants to know more complicated information from the database, it may take days or months to find out using only queries. However, it would require a much shorter time, sometimes even in minutes or hours if using data mining algorithm. Once the data mining tool has found the additional information that people need, the query environment can be used again to analyze the profiles found. KDD is not an activity that stands alone. It needs some necessary preconditions to begin data mining.

People must remove duplicate records, correct typographical errors in strings, and add missing information, etc [AdZa96]

### 4.2.2 Comparing with other Perfect Fixed Point Algorithm

Ullman described a perfect fixed point algorithm in his book, *Principles of Database and Knowledge-base system* [Ullm88]. The perfect fixed point algorithm described in this thesis is similar to that one. But, there is a difference between them. Ullman computed the union of the symbols (DOM) appearing in the EDB relations and in the rules themselves when the subgoal is negated. If a negated subgoal is $q(X_1, \ , X_n)$, let Q be the relation already computed for q (or given, if q is an EDB predicate). Let the IDB relation for this subgoal is:

DOM×…×DOM (n times) –Q

Then, for each stratum I, if a subgoal is negated, use the above relation to replace the subgoal.

However, we have a somewhat more restricted environment than the environment discussed in Ullman's book. Our environment makes it unnecessary to compute the DOM and replace the negated subgoal with DOM×…×DOM (n times) –Q for IDBs with embedded negations. When the predicate heads of rules with negated subgoals is evaluated, the IDB relations for that negated subgoal are already known because the predicates of any negated subgoal always has a lower stratum than the predicate of its head. The program checks whether the IDB relations for the predicate of the negated subgoal is known already. A practical testing of the modified algorithm works, but has not been proven correct formally. The algorithm shown in this thesis can save time when it is applied in mining data from the knowledge-base.

# CHAPTER V

# SUMMARY AND FUTURE WORK

It is attractive and challenging to extract knowledge rules from a relational database to form a knowledge-base, then mine more useful data from the knowledge-base. Mining data from a database or knowledge-base can be applied to marketing, research, and experimental designs. It is obvious that knowing and applying data mining should bring great commercial opportunities and research progress.

In this thesis, the perfect fixed point theorem is used. The perfect fixed point theorem is one of the easiest way to mine data from a knowledge base. It begins using from the facts of the database, extracting the useful data until it reaches a point that there are no more new facts that to be extracted. Also, the facts are extracted from the knowledge base without repetition. Two examples have been presented to show how the algorithm works.

Although it has been proved that by using the fixed point theorem people can mine data from the knowledge base, it still is a technology developed and practiced primarily by DBAs (Datebase Administrator) or some professionals with logic programming knowledge. Better data mining tools should be used by those who are business managers, scientists, or engineers without database management and logic programming knowledge.

Bottom-up evaluation was used in this thesis. Some research has shown that the bottom-up evaluation is more effective than the top-down one. The bottom-up evaluation can perform the evaluation in linear time. Other researchers showed that top-down evaluation is more effective with constrains. This should be researched future to find out which method is better.

There is more work to be done to apply the algorithm presented in this thesis to real life in situations where a computer layman can use the results.

## BIBLIOGRAPHY

[AdZa96] Adriaans, Pieter and Zantinge, Dolf. *Data Mining*. Addison-Wesley, 1996.

[BrMy86] Brodie, Michael L. and Mylopoulos, John. *On Knowledge Base Management System*. Spring-verlag New York Inc, 1986.

[CHY97] Chen, Ming-Syan, Han, Jiawei, Yu, and Philip S.. *Data Mining: An overview from Database Perspective, IEEE Transactions on Knowledge and Data Engineering*, 1997.

[FPSSmU96] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R.. *Advances in Knowledge Discovery and Data Mining*. P353-356. AAAI Press / The MIT Press, 1996.

[Frost86] Frost, R.A.. *Introduction to knowledge base system*. Collins Professional and Technical Books, 1986.

[Gard89] Gardarin, Georges. *Relational Database and Knowledge bases*. Addison-Wesley, 1989.

[Hugh97] Hughes Technologies. *Mini SQL 2.0 online Manual*. Internet WWW page, at URL: <http://www.Hughes.com.au/library/msql2/manual/> (version current at 10 October 1997).

[KoSi91] Korth, Henry F. and Silberschatz, Abraham. *Database System Concepts, 2nd Edition*. McGraw-Hill, Inc, 1991.

[PSFr91] Piatetsky-Shapiro, G. and Frawley, W. J.. *Knowledge Discovery in Databases*, AAAI Press / The MIT Press, 1991.

[RaSu91] Ramakrishman, Raghu, Sudarshan, S.. *Top-Down versus Bottom-Up Revisited*. ISLP 1991:321-336.

[RStSu94] Ramakrishnan, Raghu, Strvastava, Divesh, and Sudrashan, S.. *Efficient bottom-up Evaluation of logic Programs*. The State of the Art in Computer Systems and Software Engineering, ed. J. Vandwalle, Kluwer Academic Publishers, 1994.

[Salton71] Salton, Gerard. The Smart Retrieval System Experiments in Automatic Document Processing. Prentice-Hall, Inc., 1971.

[Tarau97] Tarau, Pau. *Index of Binprolog*. Internet WWW page, at URL: <http://clement.info.umoncton.ca/BinProlog> (version current at 10 October 1997).

[Ullm88] Ullman, Jeffrey D.. *Principles of Database and Knowledge-base Systems* Volume 1. Computer Science Press, Inc, 1988.

## APPENDICES

# APPENDIX A

## GLOSSARY

**API**: is called application interface. It allows the communication between programs written by different languages.

**C-ified**: is to translate the original program into C program.

**DBA**: Database Administrator.

**EDB**: A predicate whose relation is stored in the database is called an extensional database relation

**IDB**: A predicate whose relation is defined by logical rules is called an intentional database relation.

**KDD**: Techniques and tools that can transform the processed data into useful information and knowledge intelligently and automatically.

**Range-restricted program**: means safe datalog programs. In this program, all the variables in the rules are limited.

**Rectified rules**: for the rules with predicate p, if all of their heads are identical, and of the form $p(X_1,\ldots,X_n)$ for identical variables $X_1,\ldots,X_n$.

**Safe rule**: if all of a rule's variable are limited, that rule is safe.

## APPENDIX B

## EDB RELATIONS FOR POTENTIAL CUSTOMER KNOWLEDGE-BASE

```
ccust(amy_foutch,a).

ccust(anna_abbott,a).

ccust(brain_boling,b).

ccust(brain_kerler,a).

ccust(cindy_nelson,a).

ccust(cindy_stone,a).

ccust(dustin_patrick,a).

ccust(greg_trane,a).

ccust(jane_bain,a).

ccust(jeff_ford,b).

ccust(jenny_baker,a).

ccust(john_chain,b).

ccust(john_clawson,a).

ccust(judy_tree,a).

ccust(laura_clinton,b).

ccust(linda_wu,a).

ccust(mary_wagner,b).
ccust(polly_johnson,a).
ccust(ryan_west,b).
ccust(steve_laner,b).
female(alice_johnson).
female(alice_tree).
female(amy_foutch).
female(anna_abbott).
female(cindy_nelson).
female(cindy_stone).
female(diana_bain).
female(jane_bain).
female(jenny_baker).
female(jenny_barker).
female(judy_tree).
female(kim_stone).
female(laura_clinton).
female(linda_shaw).
female(linda_smith).
female(linda_wu).
female(mary_wagner).
female(mary_wayn).
female(polly_johnson).
female(sharron_Lou).
female(shiley_hook).
female(tina_shaw).
friend(amy_foutch,linda_shaw).
friend(anna_abbott,chris_lando).
friend(brain_boling,fred_grason).
friend(brain_kerler,alice_johnson).
friend(cindy_nelson,john_dean).
friend(cindy_stone,linda_smith).
friend(dustin_patrick,sharron_Lou).
friend(greg_trane,alice_tree).
friend(jane_bain,jenny_barker).
friend(jeff_ford,john_tylar).
friend(jenny_baker,diana_bain).
friend(john_chain,greg_laner).
friend(john_clawson,shiley_hook).
friend(judy_tree,tom_smith).
friend(laura_clinton,amy_foutch).
friend(linda_wu,mary_wayn).
friend(mary_wagner,tina_shaw).
friend(polly_johnson,chris_lin).
friend(ryan_west,kim_stone).
friend(steve_laner,doug_neily).
live(alice_johnson,3).
live(alice_tree,2).

live(amy_foutch,3).

live(amy_foutch,4).

live(anna_abbott,1).

live(brain_boling,2).

live(brain_kerler,1).

live(chris_lando,1).

live(chris_lin,1).

live(cindy_nelson,3).

live(cindy_stone,3).

live(diana_bain,4).

live(doug_neily,1).

live(dustin_patrick,4).

live(fred_grason,3).

live(greg_laner,2).

live(greg_trane,2).

live(jane_bain,1).

live(jeff_ford,2).

live(jenny_baker,4).

live(jenny_barker,2).

live(john_chain,2).

live(john_clawson,3).

live(john_dean,3).
live(john_tylar,2).
live(judy_tree,1).
live(kim_stone,4).
live(laura_clinton,3).
live(linda_shaw,1).
live(linda_smith,4).
live(linda_wu,1).
live(mary_wagner,4).
live(mary_wayn,1).
live(polly_johnson,1).
live(ryan_west,4).
live(sharron_Lou,2).
live(shiley_hook,1).
live(steve_laner,2).
live(tina_shaw,3).
```

# APPENDIX C

## NEW IDB RELATIONS FROM FAILY RELATION KNOWLEDGE-BASE

```
cousin(ann,tom).
cousin(doug,judy).
cousin(judy,doug).
cousin(tom,ann).
jrel(alice,ann).
jrel(alice,doug).
jrel(alice,john).
jrel(ann,alice).
jrel(ann,judy).
jrel(doug,alice).
jrel(doug,tom).
jrel(john,alice).
jrel(john,judy).
jrel(john,tom).
jrel(judy,ann).
jrel(judy,john).
jrel(tom,doug).
jrel(tom,john).
related(alice,ann).
related(alice,doug).
related(alice,john).
related(ann,alice).
related(ann,judy).
related(ann,tom).
related(doug,alice).
related(doug,judy).
related(doug,tom).
related(john,alice).
related(john,judy).
related(john,tom).
related(judy,ann).
related(judy,doug).
related(judy,john).
related(tom,ann).
related(tom,doug).
related(tom,john).
sibling(alice,john).
sibling(john,alice).
```

APPENDIX D

NEW IDB RELATIONS FROM POTETIAL CUSTOMER KNOWLEGDE-BASE

```
pcust(amy_foutch).

pcust(anna_abbott).

pcust(cindy_nelson).

pcust(cindy_stone).

pcust(diana_bain).

pcust(jane_bain).

pcust(jenny_baker).

pcust(jenny_barker).

pcust(judy_tree).

pcust(linda_shaw).

pcust(linda_smith).

pcust(linda_wu).

pcust(mary_wayn).

pcust(polly_johnson).

sbuyer(anna_abbott).

sbuyer(cindy_nelson).

sbuyer(cindy_stone).
sbuyer(jane_bain).
sbuyer(jenny_barker).
sbuyer(judy_tree).
sbuyer(linda_shaw).
sbuyer(linda_wu).
sbuyer(mary_wayn).
sbuyer(polly_johnson).
```

V

VITA

Liwei Cai

Candidate for the Degree of

Master of Science

Thesis: USE OF THE FIXED POINT THEOREM TO MINE DATA FROM A KNOWLEDGE-BASE

Major Field: Computer Science

Biographical:

Personal Data: Born in Luoyang, P. R. China, on December 20, 1971, the daughter of Guangdao Cai and Baozhen Wang. Married to Wuzi Gao in March, 1995.

Education: Graduated from No. 3 High School, Luoyang, P. R. China in July, 1988; received Bachelor of Science in Polymer Processing Machinery from Beijing Institute of Chemical Technology, Beijing, P. R. China in July, 1992. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May, 1998.

Experience: Teaching Assistant, Department of Computer Science, Oklahoma State University, January, 1997, to December, 1997; Computer Programmer, Department of Agriculture and Biosystem Engineering, Oklahoma State University, May, 1997, to December, 1997; Assistant Engineer, Liming Research Institute of Chemical Industry, P.R. China, July, 1992, to November, 1995.