Chapter: 4

A Computational Model for Knowledge Based Document Selection

- Introduction
- Components of knowledge System
- Knowledge Representation Scheme (KRS)
- Horn Clause
- Logic as Knowledge Representation Scheme
- Logic Programming
- Database Maintenance/Information (DBMI) Module
- Knowledge Base
- Knowledge Base Maintenance/Information (KBMI) Module
- Subject - Expert Model
- User Interface
- Conclusion
- References
4.1 Introduction

The advent of artificial intelligence especially the success of domain specific expert system paradigm has paved the way for many an applications of knowledge based computer system. As has been discussed in chapter 2, the rule based systems have become quite attractive as much of the knowledge and experience can be reduced to a set of rules. Especially the experience of people in different subject fields could be represent in the form of heuristics. The interaction between the knowledge engineers and the domain specific experts not only emphasis on the knowledge of the subject but also the heuristics. The present work has attempted to interact with the domain experts in the field of chemistry, i.e., the faculty of chemistry to enrich the rule based system for book selection. However, it should be noted that the survey research method warrants to identify the total popular and also an appropriate sampling method to conduct a survey. Unlike the survey method, the interaction with the domain experts does not really require a survey. The only emphasis is on the best available experts or experts whom we believe are quite knowledgeable of the task at hand. Hence only very few experts are consulted to gather the knowledge in addition to the knowledge present in the books on document selection and acquisition.

4.2 Components of Knowledge System

Since it is a knowledge based system, discussion of contents of the knowledge system is important. Three components of the knowledge system are considered as detailed below:

- Data Base
- Model Base
- Knowledge Base
4.2.1 DATABASE

It may consist of administrative data from the transactions. The system may be involved in management data about results, products, project etc. and external data of economic or statistical nature. A huge list can be drawn on the relevant data contents of the database. In case of bibliographic data, typically the knowledge base may consist of information about the following components:

1. Collection type
2. Call Number
3. Book Number
4. Author
5. Title
6. Edition
7. Imprint
8. Series
9. Notes
10. Accession Number and
11. Tracing

- Collection type include whether the material is a book, text book, conference proceedings, monograph, report, periodical, or serial publication.

- Call Number is a codification of the subject content of the book into numerical number. This is also a location mark for the book arranged on the shelves.

- Book number include first letter of the author followed by author number and first letter of the title. This helps as an individualization.
mark in the call number of a book. Many libraries follow Cutter's method in this regard.

- The author could be an individual, corporate body, etc.


- The imprint consists of place of publication, name of the publisher, and the date of publication.

- Series note contains information about the books published under a series title with individual title for each book.

- Notes part consists of contents or special features of the book.

- Accession number is a unique number assigned to each book after it is purchased and added to the library collection. This is the data normally present in the Acquisition register.

- Tracing section details different approaches a book can be searched in the library by going through the catalogue cards arranged in the catalogue cabinets. i.e., author, title, joint author, editor, subject, etc. In other words, it contains the information about the added entries.
4.2.2 MODEL BASE

This component of the knowledge system stores domain specific knowledge. The model storage functions include model representation, model abstractions, physical model storage, and logical model storage. This provides flexible and dynamic modelling components of knowledge system. Like the recent developments in data representation, the majority of the techniques for model representation make use of Artificial Intelligence Knowledge representation concepts. The content of model base can be decided on the basis of readily available models. The Problem Processing System (PPS) component of the knowledge system is envisaged to be capable of formulating appropriate models. It is not very important to identify a priori the complete set of models which constitute the Model Base.

4.2.3 KNOWLEDGE BASE

This part of the knowledge system consists qualitative information of declarative nature.

4.3 Knowledge Representation Scheme (KRS)

We have discussed in the earlier section that the knowledge system has three distinct components. But if we follow a uniform representation scheme, then the modelling knowledge, environmental knowledge, and qualitative knowledge can be integrated into a single component of knowledge system. This amounts to identifying a single knowledge representation scheme which can facilitate data as well as model representation. Bonczek et al. (1981) introduced a new database structuring that supports a unified view of modelling and environmental knowledge.
They propose a Horn Clause method for such representation. The present work the application of Horn Clause technique for the present context.

4.3.1 HORN CLAUSE

Horn Clause is a special case of formal logic approach of knowledge representation. Any other predicate calculus approach of knowledge representation can be equivalently represented by a set of Horn Clause and Horn Clause can be easily converted to an equivalent clausal form. Thus the Horn Clause is a normal form for predicate calculus approach to knowledge representation. Horn Clauses are implications of the form

\[(A_1 \land A_2 \land \ldots \land A_n) \rightarrow B\]

Where \(\land\) is the conjunction operator and \(\rightarrow\) implication. Any predicate calculus expression can be rephrased as a set of one or more such Horn Clauses. The predicate \(A_1\) is classified into three categories: (1) Static, (2) Dynamic, (3) Parametric. The static predicates represent data base component of knowledge system. The dynamic predicate represents models and its arguments correspond to formal arguments of the associated procedures. For simplicity we assume that any Horn Clause contains at most one dynamic predicate which is the right most predicate.

4.3.2 LOGIC AS KNOWLEDGE REPRESENTATION SCHEME

The first order logic facts are relations between facts and represented as logical formulae in a knowledge base. The benefits of logic as a knowledge representation scheme include:

Well Defined Semantics

The semantics of first order logic is formally well defined.
Simple Notation

The syntax of first order logic is simple and well understood.

Conceptual Economy

Each formula, fact or deduction rule is represented once independently of its different uses.

Representational Uniformity

Facts, hypotheses, implications, qualifiers, and views are all expressed in the same first order language. Even meta-level assertions can be enforced in the knowledge base, i.e., integrity constraints can be programmed (though not directly asserted) in logic.

Operational Uniformity

First order proof theory is the sole mechanism for query evaluation. As a result of representational and operational uniformity, inference rules can be directly applied to the facts.

Standard Representation Scheme

Logic provides a common basis for defining and comparing other knowledge representation schemes.

Generality of Inference and Proof Procedures

Inference and proof procedures can be used for querying, deduction of new facts, retrieval, problem solving, and theorem proving.
Defining New Logics

Once the semantics of a new logic i.e., incomplete information in time space is well understood, the semantics can be formally defined using first order logic. The definitions then extend the expressive power of the logical system.

4.4 Logic Programming

Logic programming provides a computational language for logic. Basically, it provides a question answering system in which axioms and facts form the premises of deductive inference for queries. You state or assert what is true and ask the system to draw conclusions. Logic programming is based on first order predicate calculus usually on the Horn Clause subject.

A major issue for logic programming implementation is how resolution (example unification in PROLOG) is done both correctly and efficiently. Resolution theorem proving can result in combinatorial explosions for two main reasons. First, all clauses and facts are stored in an unstructured homogenous database. Second, all clauses are treated by the theorem prover without regard for their semantics. To reduce these problems, some logic programming system provides means for expressing control of procedural knowledge.

Logic programming provides both a potential knowledge representation scheme and a computational model for intelligent databases or knowledge bases. Logic languages also provide limited means for storing and updating clauses in a knowledge base.
4.4.1 PROLOG

PROLOG is a programming language based on logic. It was invented and developed in the Artificial Intelligence community and its ability to traverse a search space efficiently and automatically has proven to be very useful there. PROLOG is a class of implementations of the positive Horn Clause subset of logic programs. PROLOG programs correspond to hypotheses. Queries correspond to theorems to be proven by the Theorem Prover, which uses unification. PROLOG is based on a procedural interpretation of positive Horn Clause in which the implication

\[ B_1 \text{ and } B_2 \text{ and } \ldots \text{ and } B_n \text{ implies } A \]

is interpreted as a procedure and reduces the problem of the form \( A \) to subproblems of

\[ B_1 \text{ and } B_2 \text{ and } \ldots \text{ and } B_n \]

Each subproblem is in turn interpreted at a procedure call to other implications. The PROLOG language is modular and flexible. Clauses in a program can be added and deleted as desired.

4.4.2 LANGUAGE FEATURES

A PROLOG program comprises a set of procedures, each of which constitutes the definition of certain predicates. A procedure consists of a sequence of clauses, which have the general form \( A \leftarrow B_1, B_2, \ldots, B_n \) to be interpreted as \( A \) (is true) if \( B_1 \) and \( B_2 \) and \( \ldots \) and \( B_n \) are true. \( A \) is said to be the head of this clause, and \( B_1, B_2, \ldots, B_n \) the subclauses. The clauses \( A \) and \( B_1 \) are examples of goals or procedure calls consisting of a predicate applied to some arguments.
A PROLOG program has primarily declarative syntax. The clauses can be read as implications universally quantified by the variables occurring in the clauses. This facilitates declarative statements of knowledge of a domain expert. It also enables declarative statements about program function, which is an important asset from the software engineering point of view.

A PROLOG program can be viewed as a high-level specification of the task to be performed. Since the semantics of PROLOG programs are consistent with logic, this enables such specifications (programs) to be manipulated in a formal fashion.

### 4.4.3 PROCEDURAL SEMANTICS

Given a query, viz., a goal to be satisfied, PROLOG tries to determine its truth in two ways. First, since facts are always true, a goal is successful (it is concluded to be true) if it matches an existing fact. The match yields a set of bindings for the variables in the goal and these bindings are available as a part of the result.

Second, a goal is concluded to be true if it matches the head $A$ of any rule $A$ if $B_1 \land B_2 \land \ldots \land B_n$ and if the sub-goals $B_1 \ldots B_n$ can be concluded to be true.

A goal consisting of a conjunction of atoms $B_1$ and $B_2 \ldots B_n$ is satisfied by first attempting to satisfy $B_1$, resulting in a substitution instance $(B_1) C_1$ where $C_1$ is a set of variable bindings. This is followed by an evaluation of the query $(B_2)$ and the remaining sub-goals.
If all of the sub goals are eventually satisfied then so is the initial goal. If the attempt to match a sub goal to the facts in the knowledge base fails and there remain alternative facts in the database, PROLOG will backtrack and try these alternatives. If all the alternatives fail then the goal is said to be unsatisfiable.

4.4.4 INFERENCE MECHANISM AND CONTROL OF SEARCH

Recall that a goal $B_1 \land B_2 \land \ldots \land B_n$ is evaluated by attempting to unify $B_1$ with the head of the rule and if the unification of the rule is successful then the system attempt to satisfy the sub goals in the body of the rule. Thus pattern matching (specially syntactic unification) and the resolution provide the basic computational model for the execution of PROLOG programs. Logically a statement is true only if it is (logically implied) by the rules and facts in the knowledge base.

The underlying computational model for PROLOG thus directly supports pattern matching (unification in particular) and backward chained inference. Furthermore, with appropriate modifications these basic mechanisms can also be used to support constraint based programming.

4.4.5 WHY PROLOG

In comparison to PROLOG, the basic formulation of most functional programming languages (e.g., Pascal, C, Ada, Lisp) lacks knowledge-based inference abilities to cope with computations involving search. In addition, there are several well-known problems for which functional solutions are either very complex or somewhat inscrutable. In comparison, the solutions
to several of these problems expressed in PROLOG are relatively elegant. This is because PROLOG can save intermediate computational resolution for subsequent use by sibling computations. The primary mechanism that facilitates this is the notion of a logical variable that enables suspended variable bindings to be dealt with in a very natural manner.

PROLOG provides for a cut operator (denoted as !) to freeze choices made since the parent goal was invoked. All other alternatives are discarded.

To conclude, the declarative syntax of PROLOG programs in conjunction with their procedural operational semantics allows for a combination of declarative and procedural representation of knowledge in a cohesive logical framework.

4.5 Implementation

The knowledge-based system is implemented in three different programming environments: CLIPPER C and PROLOG. The system consists of a database, knowledge base, model base, and user interface.

4.5.1 DATABASE

The database holds the information about books and is fairly exhaustive. The records in the database are stored in a specific format.
The database has 13 fields and adheres to the following structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Field type</th>
<th>Type</th>
<th>Width</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Group</td>
<td>Numeric</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Coll Type</td>
<td>Character</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Class No</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Book No</td>
<td>Character</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Author</td>
<td>Character</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Title</td>
<td>Character</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Edition</td>
<td>Character</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Imprint</td>
<td>Character</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Collation</td>
<td>Character</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Series</td>
<td>Character</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Notes</td>
<td>Character</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Accession No</td>
<td>Character</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Tracing</td>
<td>Character</td>
<td>150</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3.1** The Structure of Database

Since the data has been stored in this format the same structure is incorporated in our system. Modifications to this database are done via the Database Maintenance / Information (DBMI) module.

However it should be noted that building relational databases systems is very simple in PROLOG. The bibliographic information about a document can be expressed as a simple fact like the following:

```prolog
book (Group Coll_Type Class_No Book_No Author )
```
The join command of SQL can be expressed as PROLOG rules

**4.5.2 DATABASE MAINTENANCE/INFORMATION (DBMI) MODULE**

This module is generically divided into Database Maintenance (DBM) module and Library Information (LI) module.

The DBM module maintains the database by providing facilities to append, delete and edit records to/from it. This particular module has been implemented in CLIPPER programming language since it offers many powerful features for database maintenance. Besides, CLIPPER program can be effectively compiled, linked, and an executable file can be produced. This feature is very important from the standpoint of the design of the Knowledge System since CLIPPER (explained in a later chapter) which is implemented in the C language has to access the DBM/LI module. To make this possible, an executable code is essential and this, we feel, could best be done using a language providing powerful Data Base Management System features and which could also be compiled (dBASE III would have failed here) and CLIPPER was the best solution.

The library information module answers set queries on the database. The queries that can be answered were those which were considered necessary. They are:

1. To list books on a particular author
2. To list books on a particular call number
3. To list books on a particular title
Some more queries can be added to this module if the need for including them arises.

### 4.5.3 KNOWLEDGE BASE

This contains an integrated knowledge of the subject experts (a few members of the Chemistry faculty) who are consulted in the matter of selection of books. The knowledge base has been implemented in PROLOG.

The knowledge base contains facts and rules which are to be used in the inference process. The knowledge base has two components: The rule base and working memory.

The rule base contains static facts and rules which are unaltered in the consultation process. The static facts are:

- `Popular_author (name subject)`
- `Affiliation_author (name affiliation)`
- `Affiliation (affiliation)`
- `Core_area (area subject)`
- `Area_of_research (area subject)`
- `Advanced_course (course subject)`
- `Emerging_area (area subject)`
- `Keyword (word)`
- `Popular_publisher (word)`

The second component of the rule base is the set of rules from which inferences are drawn. Each rule consists of a number of conjunctive conditions and a conclusion. Conditions are predicates which may be found as facts in the knowledge base or their veracity can be determined from user input. The conclusion of a rule is a PROLOG `assert` premise which is moved to be true during the inference process.
Some of the rules and their structure are shown below

Book_value (serial_no MUST ) -
  book_weight (serial_no weight)
  weight > 66

Crit_ I_priority (serial_no FIRST ) -
  crit_ I_weight (serial_no weight)
  weight > 20

Evaluate_crit_ I (serial_no weight) -
  book (serial_no title _ _ subject _ _ _ _)
    is_core_area (title subject)
    weight = 30

Evaluate_crit_ I (serial_no weight) -
  book (serial_no title _ _ subject _ _ _ _)
    is_research_area (title subject)
    weight = 25

If it is a core _ subject or a research _ area then Subject content gets a
weight greater than 20

Evaluate_crit_ I (serial_no weight) -
  book (serial_no title _ _ subject _ _ _ _)
    is_advanced_area (title subject)
    weight = 15

Evaluate_crit_ I (serial_no weight) -
  book (serial_no title _ _ subject _ _ _ _)
    is_emerging_area (title subject)
    weight = 5

Evaluate_crit_ I (serial_no weight) -
  weight = 0

If the subject content does not suggest any one of the above areas
zero_weight is given to it

Working memory forms the dynamic component of the knowledge base At
the start of each consultation working memory is empty, but new facts are
added to it as rules are fired and inferences are made. In addition to inferred facts, user input is also stored in the working memory. A few of the dynamic facts are listed below.

- **Book**: (serial_no, title, author, type, subject, publisher, year, price, no_of_copies)

User input is read from a file and stored by asserting this fact into the database.

- **Book_weight**: (serial_no, weight)
  - This calculates total weight for a book and is stored by asserting this fact.

- **Crit_I_weight**: (serial_no, weight)
  - The weightage a book receives for its subject content. The first criteria to be tested is stored in this predicate.

- **Crit_II_weight**: (serial_no, weight)
  - The score from intellectual content is stored using this predicate for each book.

- **Crit_III_weight**: (serial_no, weight)
  - The weightage a book receives for its potential use is stored using this predicate.

- **Crit_IV_weight**: (serial_no, weight)
  - A book's score for its relation to the Library's collection is asserted using this fact.

- **Crit_V_weight**: (serial_no, weight)
The points a book scores under bibliographic consideration is stored in this predicate

4.5.4 KNOWLEDGE BASE MAINTENANCE/INFORMATION (KBMI) MODULE

The Knowledge base is maintained by the knowledge base maintenance/information (KBM/I) module. The user is provided with two options:

1. To modify the knowledge base. The knowledge base maintenance (KBM) module has to be chosen and

2. To query the knowledge base. The knowledge base Information (KBI) module has to be chosen.

To enable the user to append, delete, or edit the static facts, the asserta ( ) and the retract () predicates are used in the KBM module.

The KBI module answers the following queries on the knowledge base contents:

1. How is a given author rated?
2. What is the affiliation of a given author?
3. How is an affiliation rated?
4. Whether it is a Core _ area given subject and area?
5. Whether it is a research _ area, given subject and area?
6. Whether it is a advanced _ course given subject and area?
7. Whether it is an emerging _ area given subject and area?
8. Whether a given word [ in a title ] is listed as a keyword?
9. How is a given publisher rated?
4.6 Subject - Expert Model

We have discussed in the earlier section that if we follow a uniform representation scheme then the modelling knowledge environmental knowledge and qualitative knowledge can be integrated into a single component of the knowledge system and that Horn Clause method is ideal for such representation. PROLOG is an implementation of Horn Clause based logic programming and our expert model is implemented in this language because of the above reasons. The set of procedures of a PROLOG infoprost along with the knowledge base represents the subject expert model of the Knowledge based system.

The model of the expert is activated when the user chooses the consult expert option from the main menu. When the user chooses the above option, a C program is activated which reacts the necessary parameters of each book to be evaluated from the terminal and stores it in a DOS file. Before writing on to the file, the program validates certain parameters which need to be validated. Then, the PROLOG program infoprost reads the input from this file and evaluates the value of each book and writes this onto another DOS file apart from asserting this as a dynamic fact in the knowledge base.

4.7 User Interface

The user interface is menu-based. The opening menu is the main menu which displays the modules that are available and guides the branching to those modules. The subsequent menus after branching collect data that is required to run the chosen model. On the completion of the execution of the module, the control transfers back to the main menu. The menu system is
designed to invoke the other modules underlying it. Hence the menu controls the working of the system.

The main menu is displayed by a C program. The Driver of the system which on the user's request calls one of the following modules:

1. Consult Expert (CE) (the subject expert model) implemented in PROLOG
2. DBM/I implemented in CLIPPER and
3. KBM/I implemented in PROLOG

Once the control is passed to one of these modules, there is direct interaction between the user and that module. And only when the user quits to the main menu does the control get back to the Driver. Thus the Driver integrates the CLIPPER, PROLOG and C programming environments.

4.8 Conclusion

Acquiring the knowledge of an expert is a complex and time-consuming process. It requires good communication skill and frequent feedback. Even with the large amount of information obtained from the initial in-depth interviews with the subject experts, this system remains as a rudimentary system with incomplete knowledge of the subject domain. It is difficult for the expert to precisely describe in total the thought processes that occur in making decisions and in each piece of information used.

The model is not designed to include the optimisation module wherein each book could be tested against various budgetary conditions to determine if it is to be purchased or not. Hence, the module is for selection decision model, we take the cost of a book as a non-factor.
Price is irrelevant to make a selection decision as distinct from purchase decision. The budget should be viewed not as an influence upon the relative extent to which selection criteria are acted upon. While high cost typically results in more care being taken in making the selection decision, the priorities those items that the library must have should have or could have do not change in response to budgetary limitations; they remain the same, whether money is available or not. Further, it should be noted that subject experts generally select titles within specific budget lines; for example, new subscriptions, audio visuals, expensive titles, current books, text books. As a consequence, a costly microform collection or multi-volume set does not compete against a current book or a new journal subscription but against other possible expensive purchases. Just as high cost should not influence a selection decision, low cost or bargain prices also should be irrelevant. If a title is ranked at priority three, its ranking does not change in response to the offer of a discount. Only when two items of equal ranking are being considered for acquisition in times of financial adversity might price determine which is actually to be purchased.

These are several inputs which need further study and analysis before they can be used in the model with confidence, e.g., institutional goals, the curricula, and academic community needs. However, it is believed the model is a useful step in the direction of crystallizing various aspects of the selection process, i.e., defining and quantifying the decision process.

If a more comprehensive system for collection development can be constructed, we believe that it will justify more research efforts since it will have greater potential. The Knowledge Base may be a more effective way of presenting collection development policies. Modeling the selection process is a step towards rationalization of selection decision making.
through out a Library System If all selectors have a similar understanding of the criteria to be used and their inter relatedness then unintended disparities in collection levels can be minimized

Perhaps the greatest benefit to be derived from developing a Knowledge Based System for document selection is the requirement that we be able to describe precisely what it is that Librarians do when they select materials for their library collections Moreover by modelling selection in explicit and rational terms a library can demonstrate to its parent Institution as well as outside agencies who often view acquisitions as a bottomless pit that the misery of making choices is governed by reasonable processes
4 9 References

1 GIBBINS (Peter) Logic with PROLOG Clarendon Oxford 1988 313p

2 GOCHET (Paul) [et al.] From standard logic to logic programming John Wiley Chichester 1988 343p


4 MAIER (David) and Warren (David S) Computing with logic programming with PROLOG The Benjamin/Cummings Menlo Park California, 1988 535p

5 MALPAS (John) PROLOG a relational language and its applications Prentice Hall Englewood Cliffs New Jersey 1987 465p

6 MARCUS (Claudia) PROLOG programming applications for database systems expert systems and natural language systems Addison Wesley Massachusetts California, 1986 325p

7 O KEEFE (Richard) The craft of PROLOG MIT Cambridge 1990 387p

8 ROSS (Peter) Advanced PROLOG techniques and examples Addison Wesely Workingham England 1989 294p

10 SCHILDT (Hebert) C power user s McGraw Hill Berkeley 1988 382p

11 STERLING (Leon S) [ed] The practice of PROLOG MIT Cambridge 1990 312p

12 STERLING (Leon) and SHAPIRO (Ehud) The art of PROLOG advanced programming techniques MIT Cambridge 1987 437p


14 WIELEMAKER (Jan) SWI PROLOG 3 reference manual University of Amsterdam Amsterdam 2000 213p