DOMAIN KNOWLEDGE SPECIFICATION USING FACT SCHEMA

by

S. Parthasarathy

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

APPROVED:

C. Egyhazy, Chairman

E. Haddad

K. Triantis

May 1991

Blacksburg, Virginia

LD
5655
V555
1991
P378
C.2

DOMAIN KNOWLEDGE SPECIFICATION USING FACT SCHEMA

by

S. Parthasarathy

Committee Chairman: C. Egyhazy, Chairman
Computer Science

(ABSTRACT)

The advantages of integrating artificial intelligence (AI) technology with data base management system (DBMS) technology are widely recognized as indicated by the results from the survey of AI and data base (DB) researchers [Brodie 1988]. Various interactions between AI and DB systems have been explored and the issues have been identified in a number of articles and discussions [Brodie et al. 1986; Davis, Buchanan, and Shortcliffe 1985; Deering and Faletti 1986]. In our work, we have focused on the use of data base systems to store large number of facts and rules for a rule-based AI system.

Conventional approaches use data base systems to store facts and provide interfaces to AI systems to perform queries on the fact base. These approaches treat facts and rules separately and do not provide for storing the inference rules, which constitute the domain knowledge, in a data base. In this work, a framework is

developed to treat domain knowledge components as objects that can be stored in a data base. Our approach views domain knowledge (expressed in inference rules) as intensional interpretation of DB states supported by the fact schema. A semantic data model is chosen to specify the fact schema since these data models provide richer organization primitives than relational data models for specifying the semantics of data in the domain. This enables us to formulate domain knowledge in terms of fact schema entities. The structure of the domain knowledge, formulated thus, is used to define the schema for a data base to store these components as persistent objects.

Thus, an architecture to store facts and rules using a DBMS that implements a semantic data model is herein proposed.

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. C. Egyhazy, for the comments and support at all stages of this work. I am also grateful to the members of my committee, Dr. E. Haddad and Dr. K. Triantis, for their constructive comments on the earlier draft. To my friend, Mr. K. Vijayashankar, I am thankful for his inspired ideas and the time he generously spent discussing the project with me. Thanks also go to Ms. Rebecca Kary for her editorial and organizational help in the final stages of this thesis.

And to my wife, Chitra, and my family, I give special thanks for their loving support and patience throughout the project.

TABLE OF CONTENTS

List of Figures vii

Chapter 1 1

Introduction 1

Chapter 2 6

AI Data Bases 6
- Data Base Support for Rules 7
- Rules and Facts 10
- Data Model to Support Rule Objects 14

Chapter 3 16

Semantic Data Models 16
- Organization Primitives 16
- SDM 19
- Semantic Information Manager (SIM) 21

Chapter 4 23

Definition of Domain Knowledge Based on Fact Schema 23
- Definition of Terms Used 23
- Definition of the Data Base 25
- Definition of Concept (c) 28
- Definition of Specialized Association (s) 31
- Definition of Inference Rules in the Domain 32
- Schema for Storing Specialization Predicates 35
- Schema for Storing Inference Rules 37

Chapter 5 43

An Architecture for AI Systems 43
Schema Support for Knowledge Components 43
Architecture Definition 49
AI Task Support 52

Chapter 6 55

Conclusions 55

Bibliography 57

LIST OF FIGURES

1. Sample Schema of the ORGANIZATION Data base 24

2. Three Levels of Schema 42

3. Implementation Architecture 50

CHAPTER 1

Introduction

Integrating data base and artificial intelligence (AI) technologies has been widely acknowledged as critical for effective use of AI technology in the evolution of computing methodologies. In this work, the interaction we have pursued is data base support for rule-based AI systems. The objective of this research is to develop a framework to implement data base support for rule-based AI systems.

AI research has produced different knowledge representation (KR) paradigms and the associated reasoning mechanisms. Representing knowledge using First Order Logic (FOL) is one of the important KR paradigms. Under this representation, reasoning tasks are performed using automatic-deduction techniques. Rule-based AI systems employ this KR paradigm to encode the expert knowledge in the domain. In rule-based AI systems, the knowledge is represented using implicational well-formed formulas (wffs). Reasoning mechanism in these systems apply the rules expressed as implicational wffs to a global data base of facts. These are "direct systems" which employ natural-deduction. Since, resolution techniques are not employed, the implicational wffs are not converted into clausal form [Nilsson 1988].

These systems typically have a rule base (set of implicational wffs in original form), a fact base (global data base), and an inference procedure (direct deduction) which accesses the rule and fact bases to perform a task. Conventional approaches to implementing such systems employ various AI languages (viz., LISP, PROLOG, OPS5) which support the complex data structure requirements. The requirement to use secondary storage when the number of facts and rules reaches a high value, prompted the research into using the results from data bases to provide this support.

Sciore and Warren [1986] propose using the techniques employed in data bases (viz., indexing, caching) to enhance the implementation of PROLOG to provide secondary storage support. Shmueli, Tsur, and Zfira [1986] discuss implementing special structures to support the rule processing in PROLOG. These attempts do not explicitly use data base management systems to provide secondary storage support. Our interest in the attempts to provide secondary storage support will mainly be in the use of data base management systems. We briefly review some of the work in this area.

Conventional approaches providing secondary storage support to AI systems focus on using the data base management systems to store and manage the fact base. Data base management systems which implement the relational data model figure prominently in these attempts. The logical view of relational data bases as a set of

ground atomic formulas and the queries as theorems to be proved, is the framework employed in these attempts. Under this framework, invocation of rules in AI systems are processed as queries against the fact base. The limitations of AI languages in handling set-valued results from the queries is the principal concern of these attempts. Bocca [1988] and Nussbaum [1988] discuss strategies for implementing PROLOG to use a relational data base management system. These strategies use reduction and delayed evaluation techniques to process the rules expressed in PROLOG. Zaniola [1986] discusses modifications to PROLOG to be used as a valid query language against relational data bases. Recognizing the issues involved in efficiently processing the queries against relational data bases, Sellis and Roussopoulos [1988] propose compilation of access path schemata for all the rules, to process the queries generated when the rules are invoked. These attempts (termed as "interface-approach") focus on developing the interface process for AI languages to use data bases.

Delcambre and Etheredge [1988] propose a Relational Production Language which is similar to OPS5. This is different from approaches touched upon earlier since the framework is provided by relation data base languages. With the development of other data models, there have been attempts to use these data models to support the data base requirements of AI systems. Raschild and Su [1988] propose a Knowledge Manipulation Language (KML) which is implemented against

Object-Oriented data model. This proposal uses the encapsulation feature of these data models to attach rules to the definition of object classes. The important issue brought out by this proposal is the need to view fact and rule bases not independent of each other by preserving the relation between them.

In our work, we import the idea of structuring the rules to preserve the relations between facts and rules. Further, we extend this idea to provide data base support to store and maintain the rules as persistent objects. The basic tenet of our approach is to develop a suitable framework to model the rules to preserve the relation with the fact base. The framework that inspired us most is due to Ullman [1988]. In his work, Ullman proposes a Datalog data model in which Datalog predicates are treated as relations. The programs are built from atomic formulas which are predicate symbols with a list of arguments. The relation corresponding to the body of a rule is evaluated using standard relational operations. The substitution which makes the body of the rule true proves the rule-head with the same substitution. What appeals to us most in this work is the development of a suitable framework to use relational data bases for rule-based systems. This is significantly different from the "interface approach" taken in the attempts outlined earlier. Our objective, then, is to develop a suitable framework to provide data base support for AI systems to store and maintain facts as well as rules to preserve the relation between them.

In the next chapter, the requirements of AI systems from data bases and the conceptual issues in viewing the data bases suitably are discussed. Arguments for the use of the semantic data model are presented. In chapter 3, the Semantic Data Model (SDM) is discussed. An outline of a data base management system that implements SDM is presented with sample data base schema. In chapter 4, our view of domain knowledge in terms of fact schema entities, expressed in SDM, is formally defined and the schema to maintain domain knowledge components in a data base is outlined. In chapter 5, an architecture to implement a rule-based AI system using SDM is proposed. In chapter 6, the implementation issues and the conclusions are presented.

CHAPTER 2

AI Data Bases

The AI systems considered in this work are "consultation" systems which support diagnostic-reasoning over a fact data base using a set of production rules. An example of such a system is MYCIN developed by Davis, Buchanan, and Shortcliffe [1985]. The input to the tasks supported by these systems is the set of observations. The inference procedure uses the appropriate rules over this set of observations to arrive at the diagnosis. The reason for considering this subset of AI systems is that, these systems typically employ large, reasonably static fact and rule bases and will benefit most from data base support.

In this chapter, a brief outline of our approach is presented. The need to provide data base support for rules and the structure of rules in AI systems is first discussed. Next, the framework proposed by us, based on our view of the inter-relations that exist between rules mirroring the inter-relations between fact model entities, is outlined. Finally, semantic data model as a suitable candidate to express the inter-relations between facts and extending them to model the inter-relations between rules, is discussed.

Data Base Support for Rules

In the previous chapter, the attempts to interface AI languages to relational data base systems ("interface approach") were briefly outlined. These attempts, using the framework provided by AI languages (PROLOG), did not support a natural integration of AI and DB technologies. Brodie et al. [1986] present a discussion of the issues in integrating logic programming and data bases. The important observations made in this work are, the differences between Data base and PROLOG paradigms and the lack of structures in PROLOG to support data structure requirements to use relational data bases.

The proposals based on the framework provided by relational data bases [Delcambre and Etheredge 1988, Ullman 1988] specify new data base languages and data models. These approaches do not suffer from the issues that were encountered using "interface approach." But, both these approaches employ data bases to store just the facts. The use of data base management systems in these approaches is limited to the role of "back-end retrievers."

In our view, any attempt to use data base management systems to support AI systems should accommodate storage of both kinds of objects (facts and rules). The systems that use data base management systems to store just the facts, will require extensive recompilations or rewrites to accommodate modifications to the knowledge

component. These systems are also susceptible to changes to the fact data base schema, depending on the implementation. So, storing rule objects in a data base will render modification of the knowledge component (rules) extremely efficient and robust.

Data base management systems implement the data structures required to support the organization primitives of underlying data model. The organization of objects to be stored in data bases is specified in the schema. To store rule objects of an AI system in a data base, the structure and the inter-relations of these objects must be derived. In the next section, the organizational aspects of rule objects are discussed.

Structure of Rule Objects

Rules in AI systems are implicational wffs with the rule head predicate (LHS) implied by a predicate clause (RHS). These implicational wffs are treated as such, without any conversion to clausal form. Typically, the rule head is proved by treating the RHS predicates as subgoals to be proved. If an RHS predicate corresponds to a fact base object, it is evaluated by generating a query against the fact base. In conventional AI systems, the predicates which correspond to the fact base objects (AI

queries) are represented as typed slot-filler[1] objects. The details of how the AI systems handle the storage of these objects is discussed by Deering and Faletti [1986]. In this work, the authors also list the issues in using conventional data base systems to store these AI objects. One of the principal requirements of a data base system to store the rule objects is identified as the support for storing AI queries as data base objects. In our work, the structure of these AI queries which link an AI system to the underlying fact base is used as the starting point to derive the structure of rule objects.

Following is the list of observations that are used to define the structure of rule objects:

1. The structure of AI queries are similar to the structure of queries expressed in Query-By-Example systems. Thus, the query is a slot-filler object matching the corresponding data base object with a constraint slot for every slot variable in the data base object.

2. The rule-head is a predicate with its arguments as the slot variables. The LHS (rule-head) predicate of a rule is connected to the RHS

---

[1] Slot-filler objects are AI queries expressed as constraints on data base object attributes.

predicate clause by implication. The rule-head predicates may be the subgoals for other rule-head predicates. The structure of inter-connections of the rule objects resembles a production system with the predicates which correspond to AI queries as "Terminals."

These observations allow us to define rule-objects as slot-filler objects and predicate objects with the required inter-connections. We proceed to explore the relation of these rule objects to the fact data base objects.

Rules and Facts

A large number of published discussions and articles in the area of AI-DB integration deal with the issues in identifying optimal boundary for the interaction between AI and DB systems. In the attempts to tackle these issues, different views of knowledge relevant to using data base systems have been proposed. These views and definitions are briefly outlined, leading to our view of relation between rules and facts.

The traditional view of data bases held by DB researchers is that, data bases model the real world. This model-theoretic view of data bases treat the organization specified at the schema of a data base as the model theory and the DB states as the interpretations of the theory [Brodie and Jarke 1986]. It is the contention of the AI

researchers that data bases do not model the real world from the perspective of knowledge level since they do not provide interpretations in which individuals are identified with representations in data bases that are used in reasoning [Brodie and Manola 1988]. The proof-theoretic view held by AI researchers, on the other hand, treats the DB schema together with the DB states as a theory with a single model [Gallaire, Minker, and Nicolos 1988; Kowalski 1988; and Reiter 1988]. This view enables the AI researchers to view data base queries as theorems to be proved under the theory supported by the data base and thus view data bases as restricted form of KR. Model-theoretic view of data bases provides the update semantics for transactions by requiring the updates to preserve the truth of the theory expressed in the schema. But, the proof-theoretic view requires the data base states supported by such a model along with the model theory to be interpreted to provide support for the reasoning mechanisms. The important result brought out by this discussion is that, though data base schema in a traditional sense does provide a model theory for the organization of data, the interpretation of the tokens of this theory constitutes the knowledge components that can be used by reasoning mechanisms.

Brachman and Levesque [1986] define knowledge as propositional interpretation of a body of information. Since the theory to organize the body of information is specified in the schema of a data base, knowledge, in our view, is the interpretation of the data base schema entities which are the tokens of the model

theory and the structure of these interpretations has to correspond to the organization of fact schema entities.

To illustrate this view, let "PERSON" be a class or a relation in a data base schema with an attribute "AGE" specified. Let the domain specification of this attribute in the schema allow only numeric values for this attribute. A propositional interpretation "OLD-PERSON" may be a numeric relation that constrains the value of this attribute (as > 75). Thus, this piece of knowledge interprets the token "AGE" for all the instances allowed under "PERSON." The predicate that provides this interpretation is a general statement about all the instances of the "PERSON" class based on the attributes and inter-connections that exist between them and all other fact schema entities.

Predicates defined in this way can be supported in an organization that corresponds to the structure expressed in the fact schema. This is the boundary of interaction identified resulting in AI queries being stored as slot-filler objects which resemble the corresponding data base objects [Deering and Faletti 1986]. The rule objects in the domain modeled by the fact schema employ these predicates as "terminals" in a production system. The interpretation provided by the rule objects use the logical connectives on these "terminal" predicates and other rule objects to set up the implicational wffs. Since the variables of these predicate clauses are

eventually bound to results of the queries against the fact base, the type of the tokens used as variables in these objects should correspond to the tokens used in the fact schema.

To illustrate this point, the example provided before is extended. Let "INSURED" be another predicate similar to the one defined before which restricts an attribute "INSURED STATUS" to a value of TRUE. Let a predicate "ELIGIBLE FOR MEDICAL" be defined as a rule-head implied by a predicate clause which is the conjunction of predicates which define "OLD PERSON" and "INSURED." The type of argument token (NAME or SSNO of the PERSON) used in the predicates OLD PERSON and INSURED must be the type of the argument token used in the rule-head "ELIGIBLE FOR MEDICAL."

Also, since the rules are viewed as interpretation of the fact schema and data base states, they permit only the inferences allowed by the model specified in the schema. Thus, the rule objects are viewed as statements about the tokens used in the model theory of the fact data base which permit only the inferences allowed by the model. Viewed in this way, the theory for modeling the rule objects correspond to the organization expressed in the fact schema.

Summarizing, we outlined the views of data bases by AI and DB researchers leading to our view of knowledge components (rules) as propositional interpretation of the model theory expressed in the fact schema along with the data base states supported. We illustrated this with an example and established that the structure of AI queries correspond to the organization specified in the fact schema. We extended this view to show that the structure of rule objects should also mirror the organization expressed in the fact schema.

Data Model to Support Rule Objects

From the discussions presented in the previous section, it is quite clear that the power of the data model employed to specify the fact data base is very important in organizing rule objects. Though relational data models provide a set of sound organization primitives, the implementations do not clearly represent the organizational specifications at the schema level. Typically, the design of relational data bases is done using a semantic data model (viz., E-R Model). The entities are specified as relation tables and the associations are specified as "foreign key" specifications. The knowledge available at the schema level does not include all the inter-connections between fact schema entities and the properties of these inter-connections.

In this work, the rule objects are modeled based on the organization expressed in the fact schema. The knowledge at the schema level in a data base management system that implements a semantic data model is more complete and closer to the model intended than in relational data bases. For this reason, we consider semantic data models as better tools to implement the fact schema.

In the next chapter, the organization primitives supported by a semantic data model and an outline of a data base system that implements this model are discussed.

CHAPTER 3

Semantic Data Models

In this chapter, semantic data models are briefly outlined and a commercial data base system that implements a semantic data model is discussed. The primitives provided by semantic data models are first presented. Next, the specification of SDM [Hammer and McLeod 1981] is presented. Finally, a commercial data base management system which implements this data model is outlined.

Organization Primitives

Semantic data models provide the data base designer with modeling primitives to represent the semantics of application data to a large extent. The semantics of the modeling primitives used, implicitly support specification of integrity constraints at the schema level. This is the main improvement over the relational data models where the integrity constraints are specified, independent of the organization expressed, using functional dependencies.

A number of semantic data models have been proposed [Peckham and Maryanski 1988]. Following are the basic data modeling primitives provided by semantic data models.

1. Classification
2. Association

In the following paragraphs, these primitives and the modeling abstractions supported are discussed.

Classification

This primitive supports the aggregation abstraction. Aggregation treats atomic data which are related as a higher level entity. In the real world there are pieces of data available about the entities. This primitive allows the designer to gather all the atomic properties available on any single entity to define the way that entity exists in the real world. The atomic properties that define the entity are also referred to as the attributes of that entity. The groups of entity instances that exhibit the same properties are referred to as a class. The attributes are the tokens in the model that refer to the properties of an entity in the world. The values these attributes assume, to define an entity instance, could be data from a domain of values or set of entity instances of a different class. The values can thus be specified to mirror the

application world. The DBMS based on the SDM ensures the values of these attributes to be in the specified domain during all data manipulations.

Classification primitive supports specification of a class and the "instance-of" relation of the members of the class.

Association

This primitive establishes the set defined relations between instances of two classes. This is required to specify the domain of values for an attribute of a class as a set of member instances from the same class (recursive association) or from a different class.

This primitive also supports the generalization abstraction. Generalization is useful to form a higher order class out of similar classes to emphasize the similarity. This abstraction corresponds to the inheritance hierarchies specified by IS-A association in semantic networks where the attributes of a super-class are inherited by the members of its subclasses. Such a generalization hierarchy is specified by the classification and association primitives to group the entities with similar characteristics into a class and set up the subclasses which share the attributes of a common super class.

The primitives and the abstractions outlined above are supported by various data models. The DBMS which implements these models support a schema definition which is richer than any relational data base implementation. In this work, the SDM proposed by Hammer and McLeod [1981], is used to develop the proposed framework.

SDM

In this section, a brief outline of SDM specifications is presented. A detailed discussion of SDM is available from Hammer and McLeod [1981].

Classes in SDM model concrete objects defined using the aggregation. The attributes defined for a class are member-attributes and class-attributes. Member-attributes define the aspect of each member of the class by logically connecting them to other classes. Class-attributes are the aspects of the collective members of the class.

Subclass connections in SDM establish the inheritance hierarchy (generalization). This connection is either attribute-defined or user- controlled. Attribute-defined subclasses are established by using a predicate to define the members of the class that will participate in a subclass connection. User-controlled

subclasses are established by user specification of the members of the class that will participate in the subclass connection.

The domain of values are established by specifying the name-class from which the class-attribute values are permitted.

Member-attribute specification is implemented by using inversion or matching the value of an attribute to some members of a different class. This feature implements the association primitive explained earlier. This is one of the important modeling features in SDM. This provides multiple views of n-ary associations among the classes. The inversion allows relative view- points of an associations based on the class which is used as the perspective class to view the association.

SDM also provides for specifying at the schema level the properties of these associations as to the cardinality, uniqueness of the members associated and whether it is mandatory.

The SDM with these modeling primitives allows for the specification of the semantic structure of the application domain closer to the application that is modeled than any conventional data bases. In the next section, we briefly outline a commercial DBMS (SIM), developed by Unisys Corporation, based on the SDM.

Semantic Information Manager (SIM)

In this section, the main aspects of SDM implementation are discussed to develop the terminology used in the subsequent presentation of our work. A detailed description of SIM is available from Jagannathan [1988].

The attribute definition of SDM is implemented in SIM by classifying the attributes as Data-Valued (DVA) and Entity-Valued (EVA) attributes. Data-Valued attributes correspond to the class-attributes while Entity-Valued attributes correspond to the member-attributes.

Classes are specified using the attribute specification to define the class properties. Only user-controlled subclass hierarchy specification is supported.

The queries in SIM are expressed by selecting a perspective class to view all the other classes based on their relationship to the perspective class. If we consider the inter-class connections as directed-arcs and the classes as nodes, then a perspective class specifies the node to start from, to traverse all the arcs in the schema which are at least transitively connected to the perspective class. Selecting the perspective class implicitly determines the direction of the associational connection, between two classes, one is interested in. The attributes of the associated class are termed as the extended attributes of the perspective class.

The discussion presented above lists the key features and introduces the terminology used in rest of this work. The next chapter is devoted to the formal definition of domain knowledge based on fact schema entities expressed in SDM.

CHAPTER 4

Definition of Domain Knowledge Based on Fact Schema

In this chapter, the fact base schema is defined as a set of classes and associations in the application domain. The predicates which specify concepts and specialized associations are developed. The specification of rules of inference in the domain as combination of the concepts and specialized associations is presented. The structure of these domain knowledge components which mirror the organization of facts in the domain is used to develop the schema for storing them in a data base.

A sample data base schema (using SIM) is used for illustrations. This data base schema models an organization. The schema diagram is presented in Figure 1. All the class-attributes are not listed. The attributes we use in our examples are self-explanatory.

Definition of Terms Used

A concept c (specified as unary predicate P_c) on any data base class C is defined to be the set of those members of C that satisfy the attribute restrictions expressed to define c.

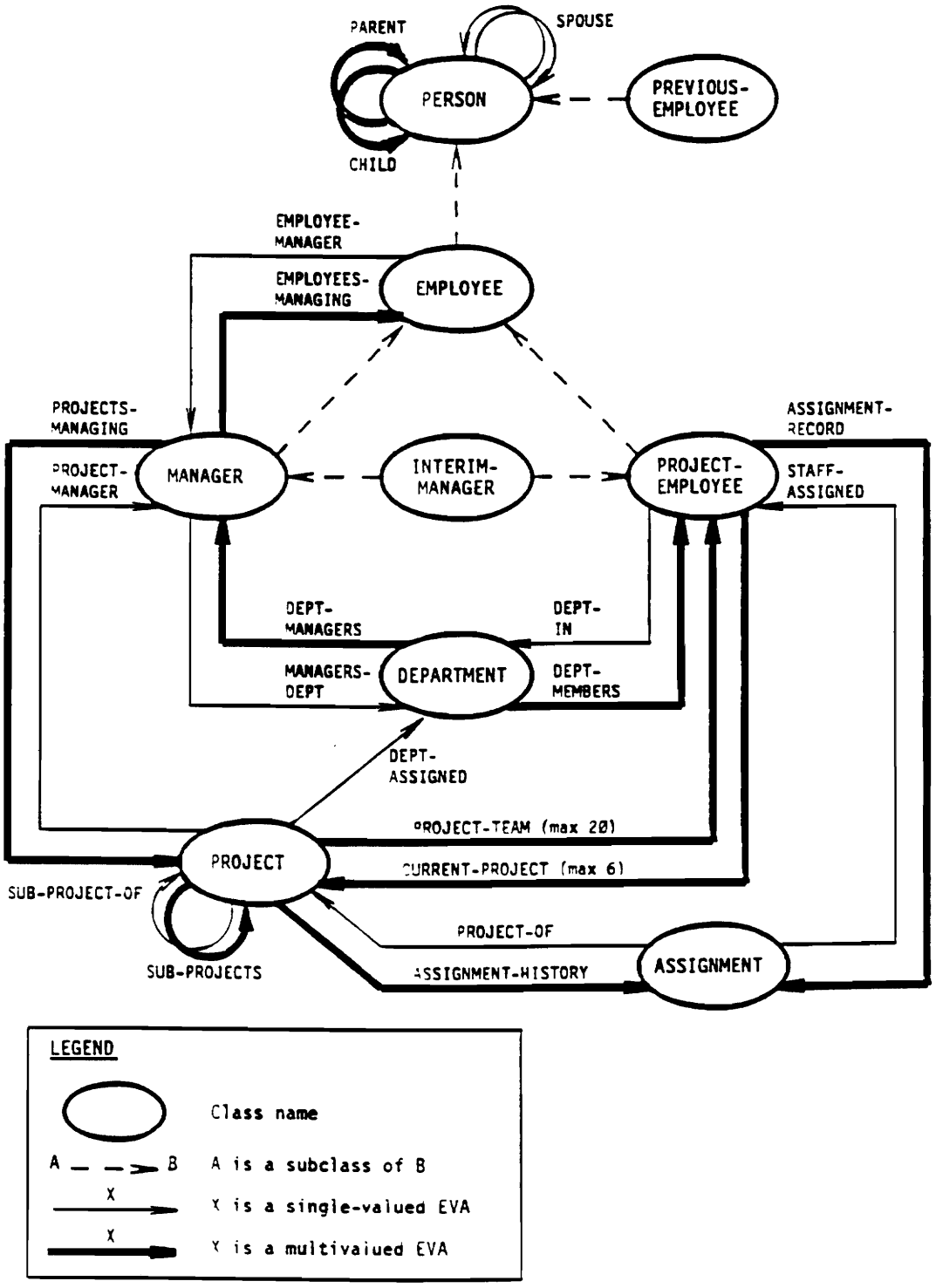

Figure 1. Sample Schema of the ORGANIZATION Data Base

A specialized association s (specified as a binary predicate P^s) is a specialization of some association P_1 between classes C_1 and C_2 in the fact schema. The members of s participate in P_1 and satisfy the attribute restrictions expressed in P^s.

An inferred concept c_1 (specified as unary predicate) is defined as a concept specified by a rule of inference (expressed in FOL).

An inferred association s_1 (specified as binary predicate) is defined as an association specified by a rule of inference (expressed in FOL).

Definition of the Data Base

A data base D is a collection of classes $C_1, C_2, \ldots C_n$ and

associations $P^l_{C_iC_j}$ where i,j <1 ... n>

and l <1 ... m>.

A class C_i is defined by the attributes $A_1^{C_i}, A_2^{C_i} \ldots A_n^{C_i}$.

For every class C_i we assume there is a sequence of attribute tokens and values X_i of the form

$(<A_1^{C_i}, V_1>, <A_2^{C_i}, V_2>, \ldots <A_k^{C_i}, V_k>)$ and $k < n$

such that,

given X_i, we can identify the member in C_i for any i

and

$P_{C_iC_j}(X_i,Y_j)$ defines a predicate relation that holds between the members identified by X_i and Y_j in C_i and C_j respectively.[2] (C_i is the perspective class.)

In our "ORGANIZATION" data base, PERSON, MANAGER, DEPARTMENT, ASSIGNMENT are the sample classes. Assigning value to the attribute token "Person-id" of PERSON will let us identify the member instance of "PERSON." The association "MANAGER-OF" between "MANAGER" and "PROJECT" is a predicate relation that holds between the members of the classes identified by "Emp-id" and "Proj-id" respectively.

Depending on the cardinality defined in the fact schema, we assume that the associations specified in the fact schema extend the attributes of a class C_i through the aggregate functions allowed on such associations.

Examples of such aggregate functions are

- COUNT of associated instances
- SUM,AVG on associated class attributes

---

[2] In our notation, the class listed first is the perspective class of the association.

which are valid only if the cardinality is > 1.

The associations also extend the attributes of a class C_i with the attributes of the associated class. Depending on the cardinality of the associations, the scope restrictions on such extended attributes are specified using quantifiers.

We denote the extended attributes as

$A^{C_i}{}_F$ evaluated by $F(P^1{}_{C_i C_j}, A_m{}^{C_j})$

where

F is one of the aggregate functions permitted or a null function with scope restriction

and

$A_m{}^{C_j}$ can be null or allowable attribute of C_j.

These assumptions are valid because the aggregate functions are independent of any particular member instance of the associated class and are not set-valued. Even though extended attributes can be set-valued, the quantifiers define a single scope for evaluation.

In our "ORGANIZATION" data base, the association "MANAGER-OF" is an association between "MANAGER" and "PROJECT" and is multi-valued in the

direction from "MANAGER" to "PROJECT." We can define following extended attribute using the aggregate function COUNT:

- COUNT of projects headed by a member of "MANAGER" class.
- PROJECT-TITLE of PROJECTS managed by a MANAGER is an extended attribute for a member instance of MANAGER class. Since this is a multi-valued association, the scope restricting quantifiers (FORALL,THERE EXISTS) reduce the scope of evaluation for these set-valued attributes.

Definition of Concept (c)

A concept which is a specialization of a general class C_i is defined by a predicate clause which restricts the values of the attributes of the class including extended attributes.

Thus the definition of a concept c (specified as a unary predicate Pc) is of the form

$$P_c(X_i) \text{ <- } OP_{k+1}(X_i, A_{k+1}{}^{C_i}, V_{k+1}) \;\&$$

.

.

$$OP_n(X_i, A_n{}^{C_i}, V_n) \;\&$$

$$OP_e(X_i, A_y, V_F)$$

where any OP_x is a valid restricting operator from the set

>,<,=,GEQ,LEQ, ... , etc., defined on A_x^{Ci}

and

A_y can be either an extended attribute A_F^{Ci} evaluated by any aggregate function F or set of extended attributes with the scope restricted by OP.

In our example, let us define a concept "BACHELOR"on the "PERSON" class as a "male person whose marital-status is single." A concept OLD as "a person whose age is greater than 75." A concept GOOD-MANAGER on MANAGER class as "a manager who got a bonus over 10000." A concept BIG-PROJECT on PROJECT class as "a project with more than 15 employees."

These can be expressed as

- BACHELOR(Person-id) <- MARITAL-STATUS(Person-id,x)
  & EQ(x,SINGLE)
  & GENDER(Person-id,y)
  & EQ(y,MALE) (1)
- OLD (Person-id) <- AGE(Person-id,x) & GTR(x,75) (2)
- GOOD-MANAGER (Employee-id) <- BONUS(Employee-id, y) &
  GTR(y,10000) (3)

• BIG-PROJECT (Proj-id) <- COUNT(Proj-id, PROJ-EMP,x) &

GTR (x,15) (4)

We can rewrite the formulas given above in the notation developed for defining concepts.

Let the classes PERSON be denoted by PE, MANAGER by MN, and PROJECT by PR.

$BACHELOR_{PE}$(<Person-id,x>) <-

EQ(<Person-id,x>,MARITAL-STATUS,"Single") &

EQ(<Person-id,x>,GENDER,"Male") (R1)

OLD_{PE}(<person-id,x>) <-

GTR(<person-id,x>,AGE,75 (R2)

$GOOD\text{-}MANAGER_{MN}$(<Employee-id,x>) <-

GTR(<Employee-id,x>,BONUS,10000) (R3)

$BIG\text{-}PROJECT_{PR}$(<proj-id,x>) <-

GTR(<proj-id,x>,COUNT(PROJ-EMP),15) (R4)

Definition of Specialized Association (s)

A specialized association is defined by a predicate clause restricting the values of the member instances of the classes which participate in an association specified in the fact schema.

Thus, the definition of a specialized association s (specified as a binary predicate Ps) is of the form

$$P_s(X_i,Y_j) \text{ <- } P_{c_i}(X_i) \ \& \ P_{c_j}(Y_j) \ \& \ P_{C_iC_j}(X_i,Y_j)$$

where

P_{c_i} and P_{c_j} are the predicates defining concepts

and

$P_{C_iC_j}$ is an association specified in the fact schema.

In our example, let us define a specialized association "BEST-COMBO," on "MANAGER-OF" association between MANAGER and PROJECT classes, as "where the project is a big project and it's [sic] manager is a good manager." We have already defined the concepts "BIG-PROJECT" and "GOOD-MANAGER." The specialized association BEST-COMBO can be expressed as

BEST-COMBO(employee-id, proj-id) <- GOOD-MANAGER(employee-id)
& BIG-PROJECT(proj-id)
& MANAGER-OF (employee-id,proj-id) (5)

We can rewrite (5) in the notation defined above as

$BEST\text{-}COMBO_{MN.PR}(<employee\text{-}id,x>,<proj\text{-}id,y>)$ <-

$GOOD\text{-}MANAGER_{MN}(<employee\text{-}id,x>)$ &

$BIG\text{-}PROJECT_{PR}(<proj\text{-}id,y>)$ &

$MANAGER\text{-}OF_{MN.PR}(<employee\text{-}id,x>,<proj\text{-}id,y>)$ (R5)

Definition of Inference Rules in the Domain

Inference rules in the domain are clauses which define inferred concepts and associations in the domain, using FOL. The inferred concepts and associations are specified as unary and binary predicates with their perspective classes in the fact domain. The clauses which define these predicates are Horn clauses of the form

P_I <- P_1 & P_2 & ... & P_n where n GEQ 1.

P_I is the LHS predicate that specifies an inferred concept or an association.

Thus, an inference rule which defines an inferred concept c_I (specified as a unary predicate) on a class C_i is written as

$P_I^{C_i}(X_i)$ <- $P_1(Z_1)$ & $P_2(Z_2)$ & ... $P_n(Z_n)$

and

an inference rule which defines an inferred association s_I (specified as a binary predicate) with C_i as the perspective class is written as

$P_I^{C_iC_j}(X_i, Y_j) <- P_1(Z_1) \& P_2(Z_2) \& \ldots P_n(Z_n)$

where $P_1 \ldots P_n$ are RHS predicates which specify inferred concept (c_I), inferred association (s_I), simple concept (c) or specialized association (s).

We restrict the inferred association to be defined between two fact schema classes C_i and C_j only if they are, at least, transitively associated in the fact schema. Inferred associations defined otherwise effectively modify the model specified in the fact schema. This restriction is also ensure the compatibility of the structure of inference rules with the model specified for the fact base.

In our example, we define an inferred concept OLD- BACHELOR on PERSON as "a person who is old and is a bachelor" using the concepts of old person and bachelor defined before.

This is expressed as

OLD-BACHELOR(Person-id) <- BACHELOR(Person-id) &

OLD(Person-id) (6)

We rewrite (6) in the notation defined above as

$OLD\text{-}BACHELOR_{PE}(<person\text{-}id,x) <-$

$BACHELOR_{PE}(<person\text{-}id,x>) \&$

$OLD_{PE}(<person\text{-}id,x>)$ (R6)

In our example, we define an inferred association MANAGER-ASSIGN, between MANAGER and ASSIGNMENT using the transitive association through PROJECT, as "an assignment managed by a specified manager if the manager manages the project to which the assignment belongs."

This is expressed as

MANAGER-ASSIGN(Assign-no,Employee-id) <-

PROJECT-OF(Assign-no,Proj-id) &

PROJECT-MANAGER (Proj-id,Employee-id) (7)

We rewrite (7) in the notation defined above as

$MANAGER\text{-}ASSIGN_{AS\ MN}(<Assign\text{-}no,x>,<Emp\text{-}id,y>)$ <-

$PROJECT\text{-}OF_{AS\ PR}(<assign\text{-}no,x>,<proj\text{-}id,z))$ &

$PROJECT\text{-}MANAGER_{PR\ MN(}<Proj\text{-}id,z>,<Emp\text{-}id,y>)$ (R7)

Having defined the inference rules in the domain in terms of the entities in the fact schema, we proceed to develop the schema to store the concepts (c), specialized associations(s), inferred concepts (c_I) and inferred associations (s_I) in a data base.

Schema for Storing Specialization Predicates

From the definitions presented above, we can easily see that concepts and specialized associations, with perspective classes, can be represented in a schema which mirrors the fact schema. We will call the schema which will represent the organization of concepts and specialized associations, "specialization schema."

Let C_i be a class in the fact schema defined by the attributes $A^{C_i}_1, A_2^{C_i} \ldots A^{C_i}_n$.

For example, in "PERSON" class, if "Age" and "Marital-Status" are the attributes, thereby declaring a class mirroring "PERSON" with "R-Age," "Age," "R-Marital Status," and "Marital-Status," we can represent the concepts on the "PERSON" class.

Let $c_1, c_2, \ldots c_m$ be the concepts specified on C_i.

These concepts are defined by predicate clauses that restrict values of the attributes of class C_i.

We represent such predicate clauses in a class C^s_i in the specialization schema with 2n attributes $R_1, A^{C_i}_1 \ldots R_n, A^{C_i}_n$

where

R_1 ... R_n are restrictive operators according to the definition of c.

Specialized associations are defined by predicate clauses that restrict values of the attributes of fact classes that participate in an association in the fact schema.

Let $P_{C_iC_j}$ be an association in the fact schema between classes C_i and C_j (with k,l attributes respectively).

Let s_1, s_2 ... s_m be the specializations defined on $P_{C_iC_j}$.

We represent such predicate clauses by mirroring the classes C_i and C_j in the specialization schema along with the association specified in the fact schema.

The fact classes C_i and C_j are mirrored as C^s_i and C^s_j with 2k and 2l attributes (according to the definition above) along with the association specified in the fact schema.

Thus, the specialization schema will consist of the same classes and associations in the fact schema. The attribute specification for these classes, whose members will be the predicate clauses, will have the restrictive operator attribute

added to each of the attributes in the Fact class. The associations will have the aggregate function attribute (if the cardinality is > 1), recursive level attribute (if the association is recursive) and the quantifier attribute (if the cardinality is > 1) declared on both the classes that participate in the association in fact schema.

Schema for Storing Inference Rules

In the definitions developed above, we have expressed inference rules as predicate clauses that define inferred concepts and associations. With these definitions, we develop a schema to store the inference rules in a data base. We call this schema which models the structure of inference rules "inference schema." We list the observations (that follow from the definitions) that are used to develop the "inference schema":

1. The inferred concepts and inferred associations are specified as predicates with perspective classes from the fact base classes.
2. The argument tokens to such predicates are the attribute tokens from the perspective class and attribute tokens from the associated classes, defined by the corresponding data type.
3. The inferred associations are between any two fact base classes (which are at least transitively associated).

4. The RHS predicate clause sets up the subgoals to be evaluated to establish the inferred concept or associations specified as a LHS predicate.
5. The "specialization predicates" will appear only in RHS predicate clauses.
6. The perspective class of RHS predicates is same as the perspective class of the LHS predicate or is a class that is associated (with the perspective class of the LHS predicate) in the fact schema.
7. In an inference rule, the RHS predicate clause is associated with the LHS predicate by the logical connective "<-" with the operational meaning "IMPLIES" or "SUB GOAL OF."
8. In the RHS predicate clause, the predicates are associated with each other by the conjunctive operator "AND."

From (1) and (2) listed above, we can easily see that the "inference schema" will have the same classes in the fact schema to group the predicates.

Let C^I_i be a class in the inference schema. The members of C^I_i are the predicates that specify inferred concepts and associations with the corresponding C_i in the fact schema as the perspective class.

Let c^I_1, c^I_2 ... c^I_n be the inferred concepts specified on a class C_i.

These inferred concepts, specified by a unary predicate, are defined by a predicate clause using the logical connectives. The arguments for the unary predicate are the attribute tokens from the sequence X_i for corresponding perspective class in the fact schema.

Let s^I_1,s^I_2 ... s^I_n be the inferred associations specified with C_i as the perspective class and C_j as the associated class.

These inferred associations, specified by a binary predicate, are defined by a predicate clause using logical connectives. The arguments for the binary predicates are attribute tokens from the sequence X_i for corresponding perspective class and the sequence Y_j for associated class in the fact schema.

So, C^I_i will have attributes to store the predicate head and the argument tokens.

From (4) we see that the members of C^I_i will include members specifying the RHS predicate clauses associated with the LHS predicate member. The member specifying a RHS predicate clause will require associations with the predicate

members which are the subgoals. Since transitive connectivity is required between the perspective class and the associated class (from (3) and (6)), the member specifying the RHS predicate will require an association with the predicate clauses that establish the subgoals in the class that is associated with its perspective class in the fact schema. From (5), since the predicate members can be "specialization predicates" the inference classes need to be associated with the corresponding specialization classes.

So, C^I_i should have an attribute specifying the member to be a LHS predicate or a member specifying the RHS predicate clause. The association expressed will mirror the associations expressed in the corresponding class C_i in the fact schema along with a recursive association and an association with the corresponding class C^s_i in the specialization schema.

From (7) and (8), the operational meaning for the recursive association and the associations mirroring the fact schema is defined as "IMPLIED BY" or "AND" depending on whether the perspective member of such an association is a LHS predicate or a member specifying the RHS predicate clause. The operational meaning for the association with the corresponding specialization class C^s_i is always "AND."

Summarizing, C^I_i is defined by the following attributes:

1. The predicate head
2. The predicate operational type (LHS or RHS).
3. The argument attributes.
4. The recursive association.
5. The associations mirroring the fact schema.
6. The association with the specialization class.

The specialization and inference schema specified above will be used to organize the knowledge component of the domain (see Figure 2). By mirroring the organization of the fact schema, we have achieved a smooth upward integration of facts to the knowledge component in the domain. The DBMS which implements the fact schema with all the fact semantics specified, is powerful enough to implement the storage structures for the knowledge component. The physical management of knowledge components will be performed by the DBMS which will make the storage and maintenance of the knowledge component transparent. By developing the operational interpretation for the associations in the inference base and specialization base, we can implement general access algorithms to support the knowledge base tasks in the domain.

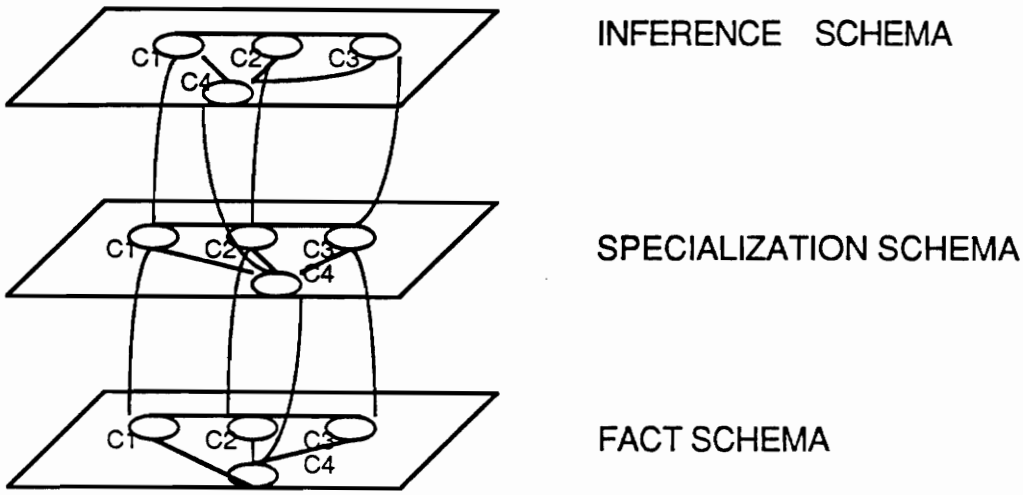

Figure 2. Three Levels of Schema

CHAPTER 5

An Architecture for AI Systems

In this chapter, the results from the previous chapter are used to propose an architecture for an AI system using a DBMS that implements a semantic data model. The specification and validation of specialization and inference schema using the fact schema and storage of specialization, inference predicates is first discussed. Next, an architecture using the DBMS, which implements the fact schema, to support the tasks performed is presented. Finally, the support for AI tasks in the domain is discussed.

Schema Support for Knowledge Components

The AI systems considered in this work consist of a large set of facts and the inference rules which are applied against this data base. The organization of this fact base is represented in the schema of the data base. We have argued for the use of a semantic data model to implement the fact schema since these data models provide for organizing domain data with the semantics specified explicitly in the schema. In the previous chapter, we formulated the domain knowledge, which consists of inference rules, based on the organizational entities of the fact schema. An outline of the organization of the knowledge components was also discussed. In this section

a detailed description of implementing the organization of these knowledge components using the data model employed to organize the fact data base is presented.

Specialization Schema

The specialization schema models the structure of specialization predicates using the primitives supported by the semantic data model. The organization specified in this schema mirrors the organization expressed in the fact schema. The class attributes for restrictive operators are added to each of the class attributes expressed in the fact schema.

In the example, let us denote the class "PERSON" in specialization schema as "S-PERSON." Similarly, other classes will be specified with a prefix "S-". Let us denote restrictive operator attribute with a prefix "ROP-" to the attribute that is defined upon. For example, the "GENDER" attribute of "PERSON" will have "ROP-GENDER" as the restrictive operator attribute in "S-PERSON."

Thus, in our example, the concepts defined on "PERSON" class will be stored in "S-PERSON" and the concepts defined on "PROJECT" class in "S_PROJECT." To store the concept "BACHELOR" in the person class, the following values for the attributes will be stored in the "S_PERSON" class.

(S-) Attribute	Value
S-PERSON-HEAD	BACHELOR
ROP-MARITAL-STATUS	EQ
S-PERSON-MARITAL-STATUS	SINGLE"
ROP-GENDER	EQ
S-PERSON-GENDER	"MALE"

The variable used for the predicates will be the key items for the class.

Similarly, to represent the concept "BIG-PROJECT" on "PROJECT,"

(S-) Attribute	Value
S-PROJECT-HEAD	BIG-PROJECT
ROP-COUNT-PROJ-EMP	GTR
COUNT-PROJ-EMP	15
S-PROJ-EMP	S-PROJECT-EMPLOYEE

where "COUNT" is a specific aggregate function attribute supported on associations with cardinality > 1 and "PROJ-EMP" is an association to the "PROJECT-EMPLOYEE" class.

The key attribute for the specialization class will be the predicate-head attribute.

The specialization schema defined above can be generated using the fact schema by employing standard naming conventions and generating restrictive attribute specifications using the attribute types specified in the fact schema.

Inference Schema

The inference schema models the structure of inference rules in the domain. The inference rules are organized as concepts and associations over the fact schema classes and associations. The classes in the inference schema mirror the classes in the fact schema along with associations. The attributes of these inference base classes and the operational interpretation of the associations have been formulated in the previous chapter.

In the example, let us denote the class "PERSON" in the inference schema as "I-PERSON." Similarly, other classes are specified with a prefix "I-". Following is the list of attributes for this class in the inference schema:

- I-PERSON-HEAD
- I-PERSON-PRED-TYPE

- I-PERSON-ARG (compound, multi-valued)
    - I-PERSON-ARG-CLASS
    - I-PERSON-ARG-ATTR
- I-PERSON-SUB-GOALS (recursive to I-PERSON)
- I-PERSON-S-PRED (association to S-PERSON)

The PRED-TYPE attribute specifies whether the class instance denotes an LHS predicate or an RHS predicate clause. The RHS type instances serve as "OR" nodes in the predicate tree which establishes the LHS predicate. The operational meaning of the recursively associated instances of RHS predicate clause is "AND."

To store the inferred concept "OLD-BACHELOR" in I-PERSON the following instances have to be supported in this class:

I-attribute	Value
I-PERSON-HEAD	OLD-BACHELOR
I-PERSON-PRED-TYPE	LHS
I-PERSON-ARG	PERSON Person-id
I-PERSON-SUB-GOALS	to OB-RHS1
I-PERSON-S-PRED	None
I-PERSON-HEAD	OB-RHS1

I-PERSON-PRED-TYPE	RHS
I-PERSON-ARG	None
I-PERSON-SUB-GOALS	None
I-PERSON-S-PRED	to BACHELOR, OLD in S-PERSON.

The inference schema defined above can be generated using the fact schema by employing standard naming conventions and generating the associative subgoal attributes mirroring the associations specified in the fact schema.

Validation of Knowledge Components

The implementation description presented above enables us to exploit the update semantics supported by the DBMS, which implements the fact schema, to validate the creation and maintenance of specialization and inferred predicate instances in the data base.

The specialization predicate instances are allowed only if the restrictive values match the data type specified in the fact schema. For example, the S-attribute of PERSON "MARITAL-STATUS" can only be of the data type defined for that attribute in PERSON.

The inference predicate instances should be validated by the maintenance procedures. The arguments specified for these predicates should be the attributes from the fact schema.

Architecture Definition

The implementation description outlined in the previous chapter provides for storing the knowledge components as persistent objects in a data base (see Figure 3). Since, the organization expressed in the specialization and inference schema mirrors the organization expressed in the fact schema, the DBMS which implements the fact schema can be used to implement the data base of knowledge components. The facts, specialization predicates and the inference rules are thus stored in a single data base.

The three main structural components of these systems are

1. Fact base layer which is the global data base.
2. Specialization layer which is the data base of predicates which can be evaluated against the fact base.
3. Inference rule layer which is the data base of inference rules in the domain.

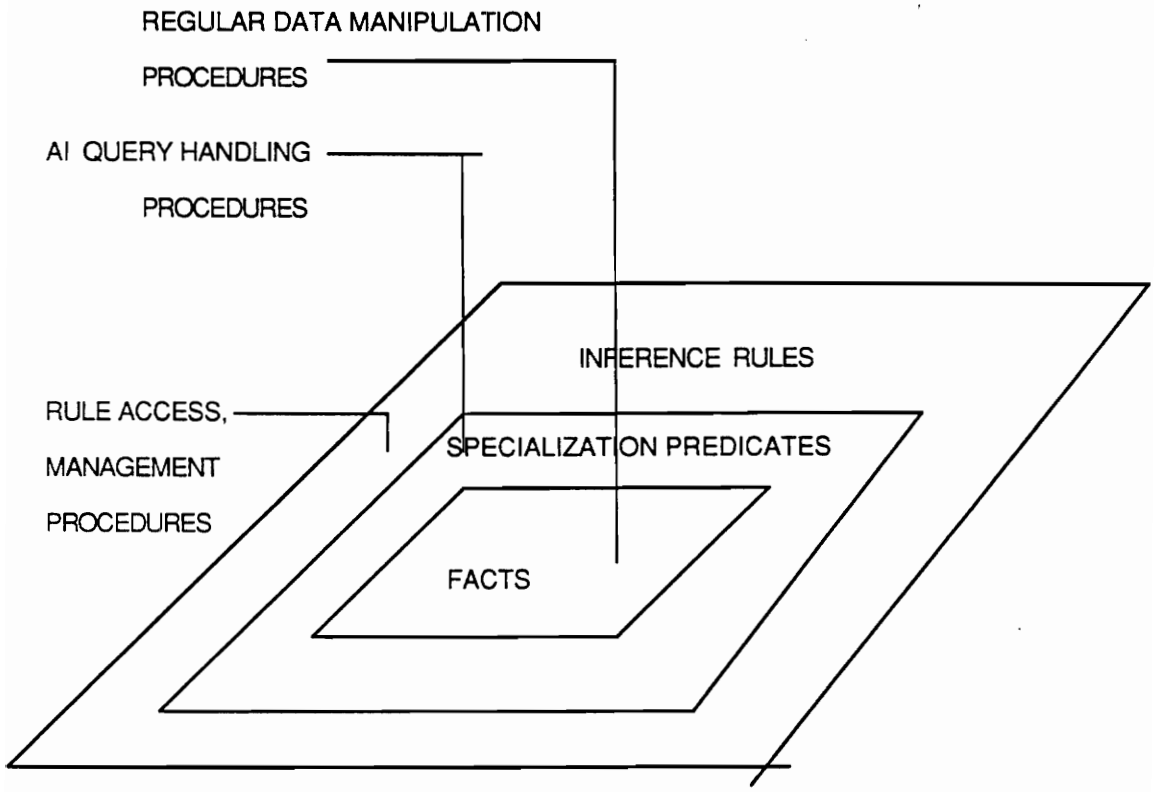

Figure 3. Implementation Architecture

The generalized processes supported in this architecture are

1. Fact base maintenance procedures which create and maintain data in the domain.
2. Specialization predicate maintenance procedures which create and maintain the specialization predicates.
3. Inference rule maintenance procedures which create and maintain inference rules in the domain.
4. Search and evaluate procedures to support the AI tasks.

The fact base maintenance procedures are regular application software to handle the flow of data from the application world. The specialization predicate and inference rule maintenance procedures are the knowledge component update procedures. They provide for updating the knowledge components to reflect changes. The access to the predicate structures maintained is provided by generalized search procedures which access the specialization predicate base and the inference rule base.

Summarizing, we have proposed a layered architecture that can be implemented using a DBMS to store the facts and knowledge components in the domain in a single data base. The knowledge components are maintained using

procedures similar to fact base maintenance procedures. The searches against these structures are supported as generalized procedures.

In the next section, we proceed to discuss the support for AI tasks provided by this architecture.

AI Task Support

The architecture proposed in this work supports the computed structures of the rules as "rule-connection graph." Since, these structures are stored as objects in the data base which is integrated to the fact base, both the top-down searches and bottom-up abduction searches are supported in this architecture.

The top-down searches can be supported by developing generalized access procedures to extract the rule-search trees from these structures. The trees extracted will have the root as the requested goal and the leaves as the queries that can be evaluated over the fact data base.

The bottom-up searches can be supported by developing procedures to use the data from the fact base and search through the rule structures stored, to derive the inferences.

These procedures can be generalized, irrespective of the content of the application. The architecture specifies the operational interpretation for the associations and the attributes specified in the specialization and the inference schema. This feature allows generalized procedures to be developed.

The access procedures could be built using the (Data Manipulation Language (DML) supported by the DBMS. After extracting the search structures, generalized evaluation procedures perform the fact base queries and pass the results up to the goal node or with the goal-predicate variable supplied the binding is performed down the tree to execute the queries at the leaves.

The variable-binding for the evaluation is supported automatically since the predicate variable data types are available from the schema definition and stored at the nodes of the search structure.

Summarizing, we have outlined the support for AI tasks provided with this architecture. The key points are

- Storage of computed structures for inferential processes
- Ease of updating these structures
- Efficient use of secondary storage through the use of a DBMS

- Generalized procedures for access and evaluation
- Support for both the top-down and bottom-up searches
- Smooth integration to the fact base.

CHAPTER 6

Conclusions

In this work, we have developed a framework for using a DBMS based on a semantic data model to support AI systems with a large number of facts and rules. We have proposed an architecture based on this framework and provided an outline of the implementation strategy. We plan to develop the algorithms for the generalized access and evaluation procedures in future. The task of designing the algorithms and implementing them will involve deciding upon the data structure support for the computed rule structures and answers extracted from the fact base, in working memory. These systems with the generalized knowledge component schema specification procedures and the search procedures can serve as a platform for implementing knowledge based systems.

With our approach, the AI systems will automatically benefit from the advances in DBMS technology (viz., distributed architecture). The generalized access procedures will generate the search space, and the evaluation procedures will support the search algorithms employed by the AI systems. Also, since the knowledge

structures are clustered, the search space is much more reduced than the search space generated without this structuring.

Finally, the issues of consistency in the specification of concepts involving a range of values will be attempted in our future work.

BIBLIOGRAPHY

Bocca, Jorge. 1988. "On the Evaluation Strategy of EDUCE." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 607-615.

Brachman, Ronald J., and Hector J. Levesque. 1986. "What Makes Knowledge Bases Knowledgeable? A View of Databases from Knowledge Level. In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 69-78.

Brodie, Michael L. 1988. "Future Intelligent Information Systems: AI and Database Working Together." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 623-673.

Brodie, Michael L., and Matthias Jarke. 1986. "On Integrating Logic Programming and Databases." In *Expert Database Systems. Proceedings on the First International Workshop*, ed. L. Kershberg. Pp. 191-207.

Brodie, Michael L., and Frank Manola. 1988. "Database Management: A Survey." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 10-34.

Brodie, Michael L., et al. 1986. "Knowledgebase Management Systems: Discussions from the Working Group. Expert Database Systems." In *Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 19-33.

Davis, Randall, Bruce Buchanan, and Edward Shortcliffe. 1985. "Production Rules as a Representation for Knowledge Based Consultation Program." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 371-387.

Deering, Michael, and Joseph Faletti. 1986. "Database Support for Storing AI Reasoning Knowledge." In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 527-648.

Delcambre, Lois M. L., and James N. Etheredge. 1988. "The Relational Production Language: A Production Language for Relational Databases." In *Expert Database Systems. Proceedings of the Second International Conference on Expert Database Systems*, ed. Larry Kerschberg. Pp. 153-162.

Gallaire, H., Jack Minker, and Jean-Marie Nicolos. 1988. "Logic and Databases: A Deductive Approach." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 232-247.

Hammer, Michael, and Dennis McLeod. 1981. "Database Description with SDM: A Semantic Data Model." *ACM Transaction on Database Systems* 6(3):351-386, September.

Hayes, Patrick J. 1985a. "The Logic of Frames." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 287-295.

Hayes, Patrick J. 1985b. "The Second Naive Physics Manifesto." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 467-485.

Jagannathan, D., et al. 1988. "SIM. A Database System Based on Semantic Data Model." In *Proceedings, ACM Conference on Management of Data*. June 1.

Kowalski, Robert. 1985. "Logic for Data Description." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 259-271.

Levesque, Hector J., and Ronald J. Brachman. 1985. "A Fundamental Tradeoff in Knowledge Representation and Reasoning." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 41-70.

Missikoff, M., and G. Wiederhold. 1986. "Towards Unified Approach for Expert and Database Systems." In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 383-400.

Moore, Robert C. 1985. "The Role of Logic in Knowledge Representation and Common Sense Reasoning." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 335-355.

Mylopoulos, John, Philip A. Bernstein, and Karry K. T. Wong. 1988. "A Language Facility for Designing Database-Intensive Applications." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 156-167.

Nilsson, Nils J. 1988. *Principles of Artificial Intelligence*. Morgan Kaufman Publishers Inc.

Nussbaum, M. 1988. "Combining Top-Down and Bottom-Up Computation in Knowledge Based Systems." In *Expert Database Systems. Proceedings of the Second International Conference on Expert Database Systems*, ed. Larry Kerschberg. Pp. 117-140.

Peckham, John, and Fred Maryanski. 1988. "Semantic Data Models." *ACM Computing Surveys* 20(3), September.

Raschild, Louiqa, and Stanley Y. Su. 1988. "A Transaction Oriented Mechanism to Control Processing in a Knowledge Base Management System." In *Expert Database Systems. Proceedings of the Second International Conference on Expert Database Systems*, ed. Larry Kerschberg. Pp. 163-174.

Reiter, Raymond. 1988. "On Closed World Databases." In *Readings in Artificial Intelligence and Databases*, eds. John Mylopoulos and Michael L. Brodie. Pp. 248-258.

Sciore, Edward, and David Scot Warren. 1986. "Towards an Integrated Database-Prolog System." In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 293-305.

Sellis, Timos K., and Nicholos Roussopoulos. 1988. "Deep Compilation of Large Rule Bases." In *Expert Database Systems. Proceedings of the Second International Conference on Expert Database Systems*, ed. Larry Kerschberg. Pp. 277-288.

Shmueli, Oded, Shalom Tsur, and Hana Zfira. 1986. "Rule Support in Prolog." In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 247-269.

Smith, John M. 1986. "Expert Database Systems: A Database Perspective, Expert Database Systems." In *Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 3-15.

Ullman, Jeffrey D. 1988. *Database and Knowledge Base Systems*. Vols. I and II. Computer Science Press.

Woods, William A. 1985. "What's in a Link: Foundations for Semantic Networks." In *Readings in Knowledge Representation*, eds. Ronald J. Brachman and Hector J. Levesque. Pp. 218-231.

Zaniola, Carlo. 1986. "Prolog: A Database Query Language for All Seasons." In *Expert Database Systems. Proceedings of the First International Workshop*, ed. L. Kershberg. Pp. 219-234.

VITA

S. Parthasarathy

Date of birth: October 15, 1958

Degrees earned:

B.S. (Applied Sciences) from Madurai University, India, 1979

B.E. (Electrical Technology and Electronics) from Indian Institute of Science, India 1982.

Professional experience:

Consultant Systems Analyst at the World Bank; Washington, D.C.; from 1986 through the present (1991).

Also worked on the SWIFT II development project in Culpeper, Virginia, through Tata Unisys Ltd., India, from 1984-86.