1988

Entity Relationship Approach to Knowledge Base Systems.

Sreerama Krishna Karukonda
Louisiana State University and Agricultural & Mechanical College

Order Number 8904544

Entity-relationship approach to knowledge base systems

Karukonda, Sreerama Krishna, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

ENTITY-RELATIONSHIP APPROACH TO KNOWLEDGE BASE SYSTEMS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by

Sreerama K. Karukonda

B.E., Andhra University, Waltair, 1979
M.Tech., Indian Institute of Technology, Kharagpur, 1981
M.S., Louisiana State University, Baton Rouge, 1986

August 1988

Acknowledgements

This dissertation is a product of my four years of pleasant educational experience at LSU. I sincerely thank the faculty and the fellow graduate students for their support and help. Foremost, I would like to thank my advisor Dr. Peter Chen for creating a very productive research environment. He always insisted on solving tough research problems and accomplishing quality work. The work presented in this dissertation is based on his seminal work on Entity-Relationship approach. I am grateful to him for giving me the freedom to experiment with new ideas. I consider myself to be very fortunate to work with him. I deeply appreciate his excellent guidance, stimulating discussions, and constant support.

I thank my committee members for their constant support, suggestions and help. Dr. Donald Kraft suggested that I should attempt a unified approach to knowledge base systems. I gratefully thank him for making me focus my attention towards this goal. His work on the unification of information retrieval models has also served as a great source of inspiration.

My special thanks go to Dr. Edward T. Lee for teaching me how to do research and encouraging me to write papers. He provided me guidance and advice whenever I needed. He encouraged me to look into some perception and computer vision problems that helped me in this dissertation.

Dr. Doris Carver made me aware of the importance of software engineering concepts. Therefore, I attempted to separate specification and implementation phases in knowledge base development. She was very kind and frequently provided me insights

on how to go about writing a dissertation. I sincerely thank her for the suggestions and the encouragement.

My interest in AI was sparked by Dr. Charles Parks. The knowledge representation background that I picked up from his courses helped me considerably in this dissertation. He provided many valuable suggestions and literature. I thank him for encouraging me to work in AI and for giving me an excellent support.

I would like to thank my peers David Cordes, Nageswara Rao, Il-Yeol Song, and Mike Stinson for the discussions and sharing many things with me.

I am grateful to the Department of Computer Science at LSU for providing me the financial assistance and the research facilities.

My parents and my brother always stood by me. I thank them for their love and affection.

I deeply appreciate my wife Vimala Devi's support, help, understanding, and love without which I could not have finished this work. She was one of my best critics and helped me with her background in psychology, phenomenology, and philosophy. I dedicate this dissertation to her.

Table of Contents

List of Figures

Abstract

A unified framework for knowledge base systems is proposed based on Entity-Relationship (ER) approach. Following the analysis and the specification of the real-world using Entity-Relationship approach, the knowledge base is implemented as a first-order logic system, a production system, or a frame-based system by mapping the appropriate symbolic data structures.

An approach for analyzing and specifying real-world perceptions must provide appropriate semantic primitives. Therefore, a justification is provided for the semantic primitives proposed in Entity-Relationship approach by considering the fundamental issues in perception. A notation that allows Entity-Relationship approach to be used as a holistic representation is presented. Translation rules are provided for the conversion of ER-diagrams into symbolic data structures of first-order logic systems, production systems, and frame-based systems. The feasibility of using Entity-Relationship approach to support a natural language front-end of a knowledge base system is examined by analyzing the representation of surface and deep structures of a sentence in Entity-Relationship approach.

CHAPTER I

Introduction

1.1. Background

This dissertation is concerned with the development of knowledge base systems. The motivation for the work described in this dissertation can be traced from the common efforts in the disciplines of data base and artificial intelligence (AI). Researchers in these two fields have been concerned with the modeling of real-world. While a number of data models are proposed by the data base community to solve this problem, different knowledge representation approaches are introduced by researchers in artificial intelligence. Computer systems developed based on data models and knowledge representation techniques are known as data base systems and knowledge base systems respectively. There is an increasing concern to integrate these approaches for the development of both data base and knowledge base systems [BMS84, BrM86, Ker87, Wie84]. This dissertation aims towards this goal.

1.2. The Problem

Although several proposals have been made for the development of knowledge base systems and data base systems, there is no unified approach to the development of knowledge base systems such as frame-based, first-order logic, and production systems, and also data base systems such as relational, hierarchical, and network data base systems. A unified approach provides a conceptual framework under which a spectrum of data and knowledge base systems can be developed.

The problem of providing a unified view to both data base and knowledge base systems can be approached in three different ways. The first one is to start with an approach that provides a unified view to data base systems and then to expand it to knowledge base systems also. The second alternative is to start with an approach that provides a unified view to knowledge base systems then to expand it to data base systems also. A third alternative is a "revolutionary" approach which is not based on any of the existing approaches.

The first alternative seems feasible, since there is a unified framework provided by the Entity-Relationship (ER) approach to data base systems [Che76]. Furthermore, recent work indicates the suitability of Entity-Relationship approach to address knowledge representation issues [FeF85, HeC85, KWD86, Laz87, Lee85, SeS85]. In the absence of a unified approach to different types of knowledge base systems, the second alternative does not appear to be an attractive one. Since the third alternative is "revolutionary", it is not known at the present time.

1.3. Our Approach

In this dissertation, we choose the first alternative and examine its feasibility by considering Entity-Relationship approach. Chen [Che76, Che77a, Che77b] showed that Entity-Relationship approach provides a unified view to data base systems based on relational, network, hierarchical, and entity set data models. Therefore, we will show that Entity-Relationship approach can also be used for the development of knowledge base systems based on first-order logic, production, and frame representations.

The approach we take to the development of knowledge base systems is divided into two phases: one for the specification and the other for the implementation. In the first phase, analysis and specification of the world is performed using Entity-Relationship approach. The specifications obtained in phase one are then implemented in phase two by mapping them onto symbolic data structures of a first-order logic system, a production system, or a frame-based system.

In addition to incorporating the concepts of software engineering into knowledge base development, this approach closely corresponds to knowledge and symbol levels proposed by Newel [New82]. In our approach, the analysis and the specification by Entity-Relationship approach is done at the knowledge level, while the implementation of the knowledge base is accomplished at the symbol level by choosing appropriate symbolic structures.

Any approach for analyzing and specifying real-world perceptions must provide proper semantic primitives. Therefore, a justification is provided for the semantic primitives proposed in Entity-Relationship approach by considering fundamental issues in perception. Perception is known to be a holistic phenomenon [AsS88, ASH87, BDL79, Kof63, MKR82], where things are perceived as wholes that stand out of their background. A whole emerges out of its general background due to the organization of its parts. Therefore, we propose a notation that allows the Entity-Relationship approach to be used as a holistic representation. This notation also provides a solution to the criticism that Entity-Relationship approach does not handle complex descriptions easily [BPR88].

The primitives normally used in Entity-Relationship approach include *entities, relationships, attributes,* and *values* [Che76, KoS85, Ull82]. The Entity-Relationship approach used in this dissertation adds the semantic primitive, *role*, to this core set of primitives in providing a holistic representation.

In order to implement the specifications provided by Entity-Relationship approach as a first-order logic system, a production system, or a frame-based system, appropriate translation rules for the conversion of ER-diagrams are provided. The symbolic data structures obtained form the translation process are then fed to a knowledge base mangement system capable of maintaining the first-order logic system, the production system, or the frame-based system implemented.

Knowledge base systems often provide a front-end that permits users to communicate with the knowledge base in a natural language. Since sentences in natural language have surface and deep structures, we show how the sentence structures can be represented in Entity-Relationship approach.

1.4. Objectives of the Dissertation

The major objectives of this dissertation can be summarized as follows:

1. Propose a two-phase development of a knowledge base system in which (a) analysis and specification of the world is performed using Entity-Relationship approach, and (b) implementation of the knowledge base is accomplished by mapping the specifications onto appropriate symbolic data structures.
2. Develop a basis for using Entity-Relationship approach for the specification of real-world perceptions. Provide a justification for the semantic primitives pro-

posed in Entity-Relationship approach by considering fundamental issues in perception. Also, propose a notation that allows Entity-Relationship approach to be used as a holistic representation.

3. Provide translation rules for the implementation of first-order logic systems, production systems, and frame-based systems from the specifications provided by ER-diagrams.

4. Show the representation of surface and deep structures of a sentence in Entity-Relationship approach to support a natural language user front-end of a knowledge base system.

1.5. Organization of the Dissertation

The current chapter briefly described the background, the problem, the approach, the summary of objectives, and the organization of the dissertation. Chapter 2 develops a basis for the specification of the real-world perceptions using Entity-Relationship approach. A justification for the semantic primitives proposed in Entity-Relationship approach is provided by considering some fundamental issues in perception. It also provides a notation for using Entity-Relationship approach as a holistic approach. Chapter 3 shows how the specifications provided by Entity-Relationship approach can be implemented as a first-order logic system or a production system. The implementation of a frame-based system from ER-diagrams is discussed in Chapter 4. The possibility of using Entity-Relationship approach to provide a natural language front-end is explored in Chapter 5 by showing how surface and deep structures of a sentence can be represented in Entity-Relationship approach. Conclusions and the directions for future research are provided in Chapter 6.

CHAPTER II

Entity-Relationship Approach

As A Holistic Representation

2.1. Introduction

Any approach for analyzing and specifying real-world perceptions must provide appropriate semantic primitives. In this chapter, we attempt to provide a justification for the semantic primitives proposed in Entity-Relationship approach by considering the fundamental issues of perception. Gestalt Psychology is one of the most influential approaches to perception [AsS88, ASH87, BDL79, MKR82, Zus70]. Therefore, it is used in our justification of the semantic primitives in Entity-Relationship approach. Our aim is to develop a basis for a holistic representation of the world using Entity-Relationship approach.

2.2. Semantic Primitives

An important issue in representation is the primitives that must be supported. Different approaches propose different set of primitives. Semantic nets use *nodes* and *arcs*. Frames are based on *properties* and *values*. Entity-Relationship approach used in this dissertation proposes *entities*, *relationships*, *roles*, *attributes*, and *values*. Predicate calculus is based on *constants*, *variables*, *terms*, *predicates*, and *conjunctions*. Although predicate calculus can serve as a useful representation, we take the view that it is a formal system of representation which is not governed by the principles of human perception. Therefore, we compare semantic nets, frames, and Entity-

Relationship approach to determine which one has a more effective set of semantic primitives. Consider the representation of the following example: the red colored ball broke the glass window. This example is represented by semantic nets, frames, and Entity-Relationship approach as shown in figures 2.1, 2.2, and 2.3 respectively.

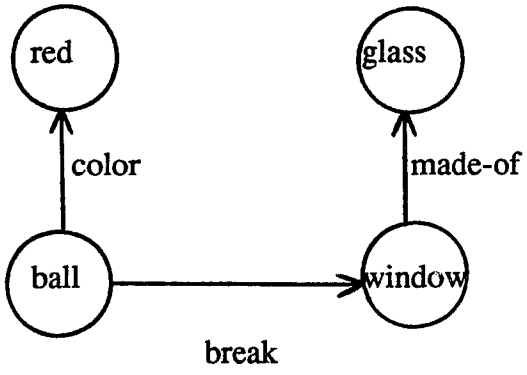


Figure 2.1. An example of a semantic net

```
(ball
        (color      (red))
        (break      (window))
)

(window
        (made-of    (glass))
)
```

Figure 2.2. An example of frames

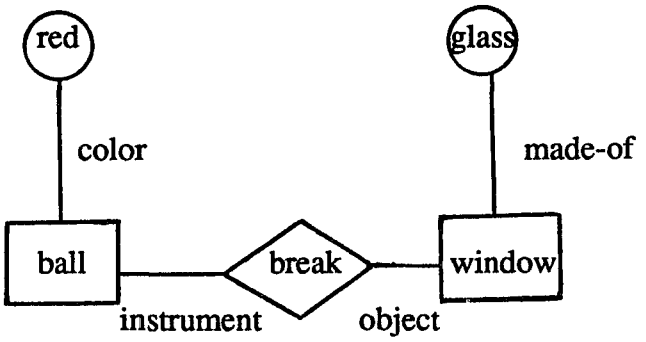


Figure 2.3. An example of an ER-diagram

The above example indicates that the semantic primitives provided by the Entity-Relationship approach are more fine-grained than the semantic primitives provided by a semantic net or a frame. While Entity-Relationship approach maintains a distinction between entities, relationships, roles, attributes and values, a semantic net treats entities and values as nodes, roles and attributes as arcs, and relationships as either nodes (in the case of n-ary relationships) or arcs (in the case of binary relationships). Similarly, in a frame-based representation, entities and n-ary relationships are represented by a frame, binary relationships, attributes, and roles are represented by slots, and values correspond to values themselves. If we show that the distinction between the semantic primitives proposed by Entity-Relationship approach is necessary, then Entity-Relationship approach becomes a more semantically expressive approach, which may then be used to guide the design of different styles of knowledge base systems.

2.3. Need for a Holistic Representation

One of the major criticisms of structural approaches such as frames, and semantic nets is that they treat a concept as a mere sum of its parts. Gestalt psychologists, however, demonstrated convincingly that a *whole* is different from the sum of its parts. Their view is that cognitive processes like perception, thinking, and learning are governed by some organizational principles which give meaning to things in the real-world [Kat50, Kof63, Koh47, KuP81, Pol69, Wer71]. For example, a train is not perceived as a heap of freight cars and an engine. A train is viewed as a *holistic* object having a linear organization of freight cars usually headed by the engine. Since our perception is a holistic phenomenon, there is a strong need for a holistic representation that can capture real-world perceptions effectively. In the following sections, we consider some fundamental issues of perception dealt in Gestalt Psychology to justify the need for the semantic primitives in Entity-Relationship approach and then we propose a notation that allows Entity-Relationship approach to be used as a holistic representation.

2.4. Perception of a Whole

As there is a psychological evidence that we perceive things as wholes, let us analyze what they are and how they are formed.

2.4.1. Figure and Ground

How does an agent perceive things ? We see things around us standing out from their background. Pictures hang *on* a wall. Words are seen *on* a page. In each of these cases, the pictures and words are the figure, while the wall and the page are the back-

ground. Therefore, the ability to distinguish an object as a whole from its general background is fundamental to all perception [MKR82].

2.4.2. Role of Attention

Although the distinction between figure and ground is an important one, an agent does not perceive everything that appears as a figure. Perception is selective and it is directed by the agent's focus of attention. Of many distinctly observable things, the agent attends only to the things that interest it.

Wholes may be simple or complex. Simple wholes are those that have no parts. Complex wholes are those that are made up of other things as their parts. When we focus our attention on any thing, whether it is a simple or complex, we view it as a single whole. We pay attention to how the thing as a whole stands out of its background, instead of how it is made up of or what its parts are. That is we are interested in the interaction between the figure and the ground than the figure itself or the ground itself. Consider a simple thing such as a dot. We perceive it as a whole only when we see it against some background such as a sheet of paper. How well we perceive an object depends on how it contrasts its background. The interplay between the object and its background gives the object the distinct characteristics that make us recognize the object. For example, how distinctly we see a dot on a sheet of paper depends on its color contrast with its background. Another characteristic that makes the dot distinct is its size referring to the extent it dominates its background, the page. We normally describe things by the characteristics that make them look distinct from their background. These characteristics are called by several names such as properties, and

features.

2.4.3. Entities as Wholes of Interest

Consider the definition of an entity as *a thing of interest perceived in somebody's mind.* In the present context, we assume an agent to represent the somebody's mind. For something to qualify as an entity, first it should be a thing, and then it should be of interest to the agent. Since we perceive things as wholes against a background, the background is usually provided by the domain that the agent is dealing with. Things are perceived only if they are important to the domain. Not all things in a domain may be of interest to the agent. Therefore, entities are only the things that are of interest to the agent in the domain.

2.4.4. Attributes of an Entity

How are entities described ? We may consider entities as the wholes that are observable against a background. We have seen in section 2.4.2 that a whole is associated with characteristics that make it distinct from its background. We introduce *attributes* to represent those characteristics or properties that make an entity to be seen distinct in its domain. Thus, attributes serve the function of describing entities. For example, the entity representing a dot on a sheet of paper is described by its attributes color, size, and shape as shown in figure 2.4.

2.4.5. Need for Values

When we notice the characteristics of a thing, we want to judge how effective they are in making the thing prominently seen. To represent the result of our

judgement of the characteristics of a thing, we introduce *values*. Consider color as an attribute of the dot in the previous example. We are interested in judging how intensely it makes the dot contrast its background, the page, to become prominent. Therefore, color as a feature of interaction between the dot and its background can assume any value within a range of possible outcomes. The specific value that color takes on depends on a given situation. We call the possible range of values that an attribute may assume as its domain of values.

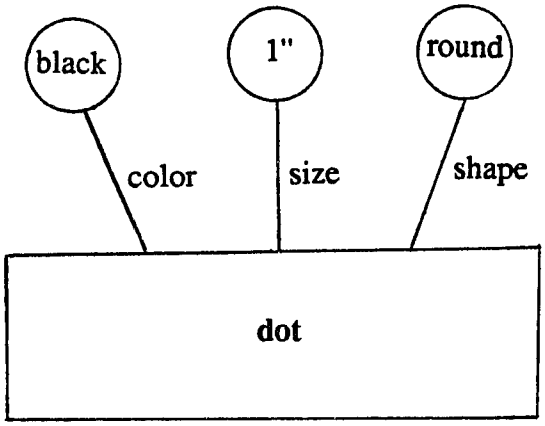


Figure 2.4. ER-diagram for the dot example

The characteristics of things may assume either qualitative or quantitative values. Color as an attribute takes on qualitative values such as red, orange, yellow, blue, and green. As an example of an attribute that takes on quantitative values is the weight of things which assumes values such as 2lb. Sometimes, our judgement of the same characteristic may be either qualitative or quantitative. For example, to simplify our judgements, we use qualitative values for weight of things such as heavy, and light.

Therefore, what an attribute relates an entity with is a value that measures the intensity or the degree with which the characteristic represented by the attribute makes the entity stand out from its background.

2.5. Organization of a Whole

We see many things in real-life that are made up of other things. A complex object such as a table is formed out of its parts: the top and the four legs. Although a table is made up of these parts, it is not a mere collection of them. A table cannot be formed without a proper organization of its parts. Gestalt psychologists emphasized that it is the organizational aspects that make the parts to merge into a single meaningful whole. The properties of the whole that emerges from its parts may have new and different properties from those of its parts.

In the following subsections of this section, we will examine the nature of parts, the process of integration of parts to form a whole, and the emergence of a whole. We assume that the perceptual agent can shift its attention either from the whole to the parts, or from the parts to the whole.

2.5.1. Parts of a Whole

When the agent shifts its attention from wholes to parts, its focus is on the parts. Parts are the things that can exist in isolation from their whole. For example, consider a table made up of its parts: the top and the four legs. Each of these parts can exist in isolation whether the table that they make up exists or not. When we see these parts in isolation we no longer see them as functional parts as the top and the legs. Instead, we

view them as a rectangular block (rb1) and four rectangular prisms (rp1, rp2, rp3, rp4) as shown in figure 2.5. When the focus of attention shifts to a particular part, it becomes the whole that is being perceived. As we saw in section 2.4.2, every whole has characteristics that make it perceivable. Therefore, when viewed in themselves, individual parts become entities which are described by their own attributes as shown in figure 2.6.

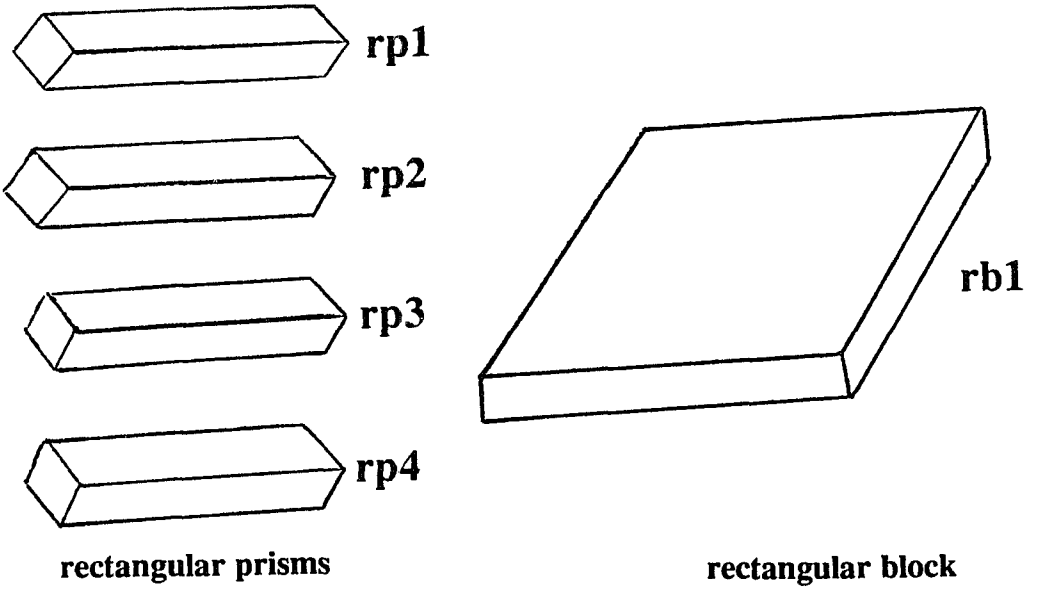


Figure 2.5. Isolated parts

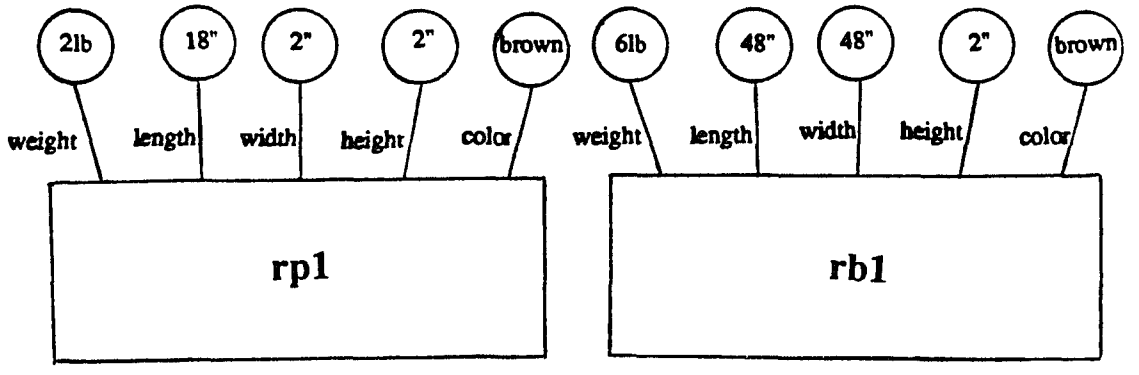


Figure 2.6. ER-diagrams for isolated parts

2.5.1.1. Structure and Function

When we see a part in isolation, it does not have any functional significance. A part assumes functional significance only when it participates in a whole. Thus a part can serve different functions depending on the whole in which it participates. If we consider the part in isolation as the structure of the thing being represented, the functionality of the thing deals with the uses of the structure. Consider, for example, the structure the rectangular prism (rp1) shown in figure 2.5. The rectangular prism, rp1, can serve many useful functions such as supporting things as in case of a table, hoisting a flag, or plugging a hole.

2.5.1.2. Roles of an Entity

To maintain the distinction between the structure and the function of things, we introduce *roles* of an entity. While attributes describe the structure of an entity, the roles describe the useful functions that the entity is capable of performing.

2.5.2. Integration of Parts

Assume that the focus of attention of the agent now shifted from individual parts to the organization that integrates them into a single whole.

2.5.2.1. Relationships between Entities

What do we mean by the organization of parts into a whole ? Suppose we want to make a specific table 'table24' out of the parts in figure 2.5. By keeping them as a simple collection as a heap in figure 2.7, they do not form a table.

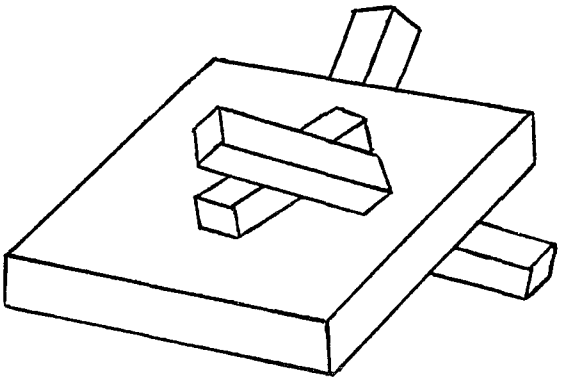


Figure 2.7. A heap of parts

We need to make sure that the parts relate to one another in a particular fashion to form the table24. It is these relationships between the parts that provide the organization needed to form a whole. Therefore, we introduce *relationships* between entities to represent how parts relate to one another. In order to form the table24, the four rectangular prisms (rp1, rp2, rp3, rp4) should support the rectangular block (rb1) as in figure 2.8. So the rectangular block participates with each rectangular prism in a separate relationship *support.*

In order to explicitly represent who supports whom, we introduce *roles* for each entity participating in a relationship. Thus, rectangular prism, rp1, and rectangular block, rb1, play the roles: *supporter* and *supportee* respectively in the relationship *support1*. An ER-diagram representing the organization of the table24 is shown in figure 2.9.

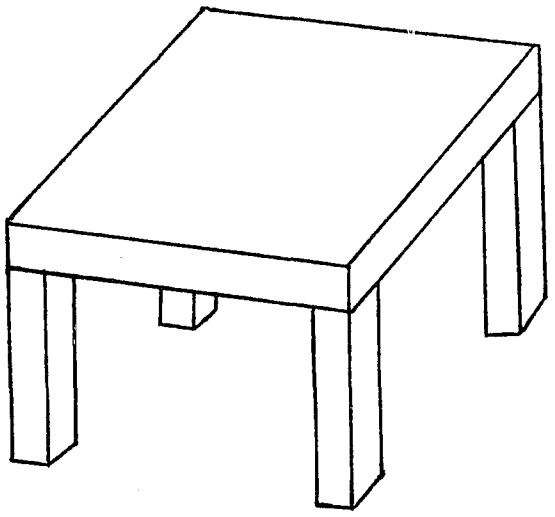


Figure 2.8. Table as an organized whole

Except in the case of very simple wholes, the organization of a whole usually involves more than one relationship between the parts. In some complex wholes, the number of relationships between the entities may be fairly large.

2.5.2.2. Attributes of a Relationship

Often relationships are judged based on some criteria such as how sound they are, how effective they are and so forth. For example, it may be important that the support provided by each rectangular prism to the rectangular block in figure 2.9 to be strong enough to form the table24. Therefore, we introduce *attributes* to relationships for representing the characteristics that describe how they relate entities. As noted in section 2.4.5, values represent the result of our judgement of a characteristic. Therefore, the value of a relationship attribute can be used to rank a characteristic of the relationship.

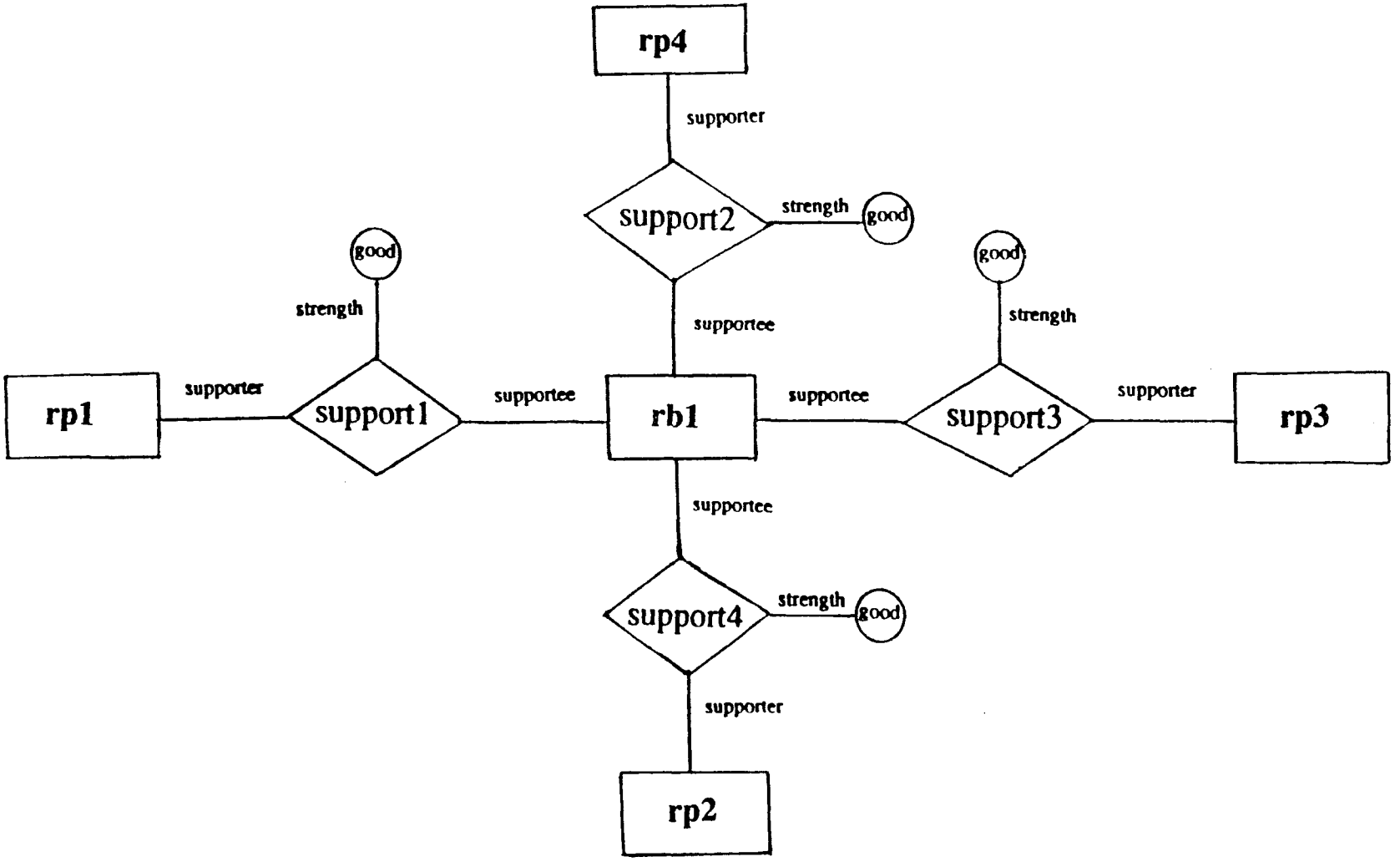


Figure 2.9. An ER-diagram showing the organization of parts in a table

2.5.3. Emergence of a Whole

Now assume that the agent shifted its focus of attention to the entire organization of parts described in section 2.5.2 as a figure against some background. Consider the agent seeing the table24 against a background such as a study room or a restaurant. The parts of the table will no longer be seen distinctly. They merge with one another to emerge as a whole, the table. Thus, the table is seen as one single integral form. Figure 2.10 shows an ER-diagram representing a table as a whole.

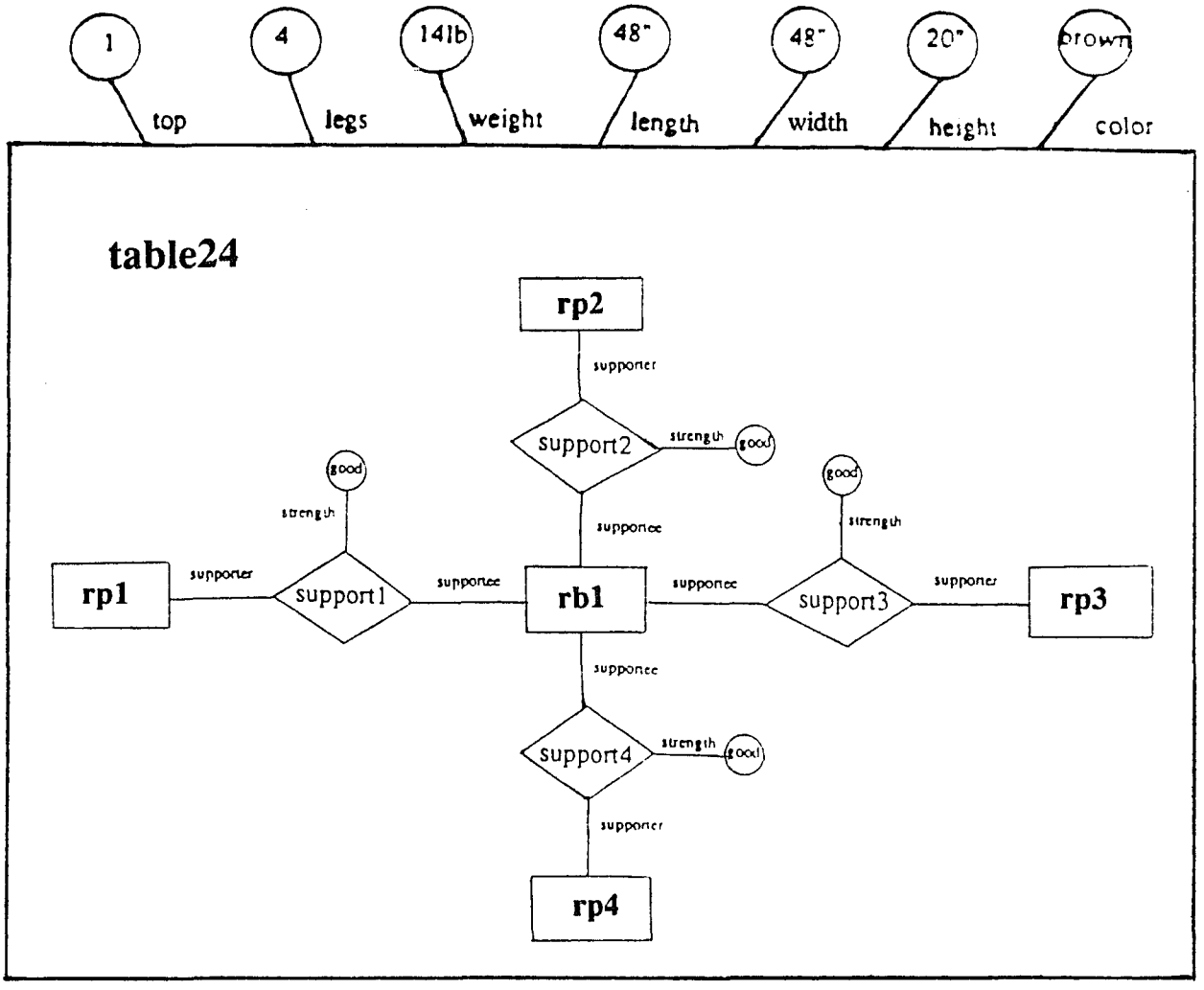


Figure 2.10. An ER-diagram representing table as a whole

We may note that the properties of a whole may have new and different properties from that of its parts. For example, the table as a whole has an integral composite shape described by the properties (or attributes) such as *top* and *legs* which none of its parts have. While the weight of the table is obtained by summing up the weights of the parts, the breadth and the length are the same as that of the rectangular block. The height of the table is obtained by the sum of the length of a rectangular prism and the height of the rectangular block. It is important to note that what kind of properties (and their values) that the whole will have depend on how its parts interact or organize with one another.

It is possible for a whole to become a part of another whole. Consider the table24 as a part of some study-room2. The table24 serves the useful role of being an 'instrument' in keeping objects such as books. The ER-diagram in figure 2.11 shows the table24 as a part of the study-room2. The parts and wholes analysis dealt here provides a semantic theory to describe complex entities formed out of other entities.

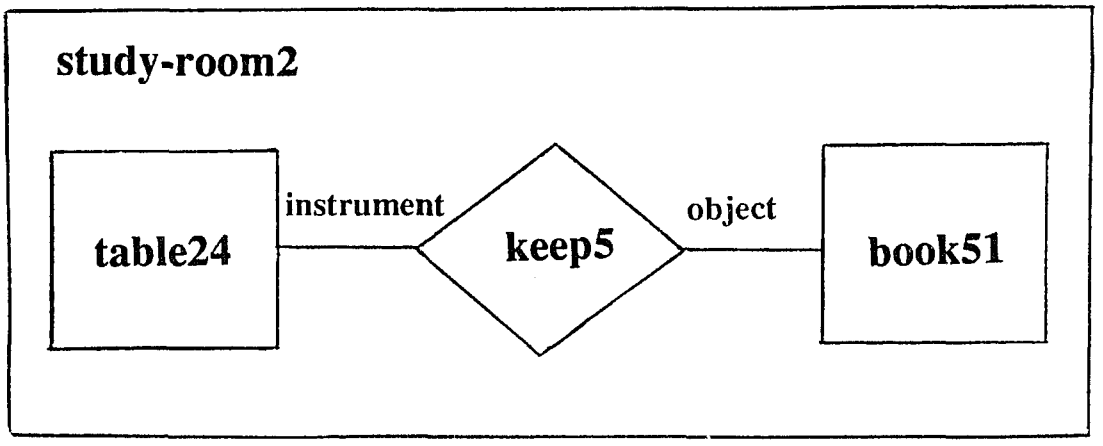


Figure 2.11. Table as a part of another whole

2.6. An ER-Notation for a Holistic Representation

Our discussion in the preceding sections indicated that entities, relationships, roles, attributes, and values are distinct from one another. It also provided a justification to the semantic primitives in Entity-Relationship approach to capture real-world perceptions. Therefore, Entity-Relationship approach can now be used to guide the semantics of other approaches in which this distinction is not clear.

We now develop a notation to represent our parts and wholes analysis using Entity-Relationship approach. This notation will be used in our attempt to implement the knowledge base as a first-order logic system, or a production system, or a frame-based system in chapters 3 and 4.

2.6.1. Entities

According to our discussion in this chapter, entities represent wholes that are of interest to us. Since we can focus our attention on an entity as a whole or on the organization of its parts, we divide the description of an entity into two components: (1) external, and (2) internal descriptions. The external description describes how an entity is perceived when viewed as a whole. The internal description deals with how the parts of the entity are organized.

Consider the table example again. In the external description of the 'table24' shown in figures 2.10 and 2.11, we are concerned with (a) its attributes, (b) its roles, and (c) the wholes in which it is a part. The attributes of table24 in figure 2.10 represent its properties such as the number of legs and tops it has, and its length, width, and height. The roles of table24 in figure 2.11 indicate the functions that it

plays in the relationships in which it is participating such as being an instrument in the relationship 'keep'. We introduce *w-entities* of an entity to indicate the wholes (i.e. other entities) in which it is a part. The table24 in figure 2.11 has the study-room2 in its 'w-entities' description. While 'w-entities' of an entity relate it with other entities, the attributes and the roles of the entity associate it with values and relationships respectively.

The internal description of the table24 shown in figure 2.9 is concerned with (a) its parts and (b) the relationships among its parts that make it up. We introduce *p-entities* of an entity to represent the entities that are its parts. The table24 has one rectangular block (rb1) and four rectangular prisms (rp1, rp2, rp3, rp4) as its p-entities. To represent the relationships that organize the parts of an entity, we introduce *o-relationships*. The four relationships 'support1', 'support2', 'support3', 'support4' are the o-relationships of the entity 'table24'.

The external and internal descriptions of an entity may be summarized as the following:

Entity:

external:

attributes	values
roles	relationships
w-entities	entities

internal:

p-entities	entities
o-relationships	relationships

The external and internal descriptions of an entity may be represented in an ER-diagram by combining its external and internal descriptions as in figure 2.12. Figure 2.12 representing the external and internal descriptions of a table is obtained when the figures 2.10 and 2.11, and 2.9 representing its external and internal descriptions are combined.

2.6.2. Relationships

We have seen that relationships play the central role in the organization of entities in forming other entities. Since relationships have a fairly complex description, we provide a separate description for relationships also. In describing a relationship, we must first indicate the whole(s) that the relationship helps organize. For example, 'support1' relationship in figure 2.9 provides organization to table24 by relating its parts rectangular block, rb1, and rectangular prism, rp1. We introduce *o-entities* of a relationship to indicate the entities that are formed by the organization provided by the relationship.

A relationship should also explicitly state the roles performed by each of the entities (i.e. the parts) participating in it. The relationship 'support1', for example, explicitly indicates that the rectangular prism, rp1, and the rectangular block, rb1, play the 'supporter' and 'supportee' roles respectively.

The description of a relationship must also include the information about its own attributes describing itself. The relationship 'support1' has an attribute 'strength' to indicate how strong the relationship 'support1' itself is.

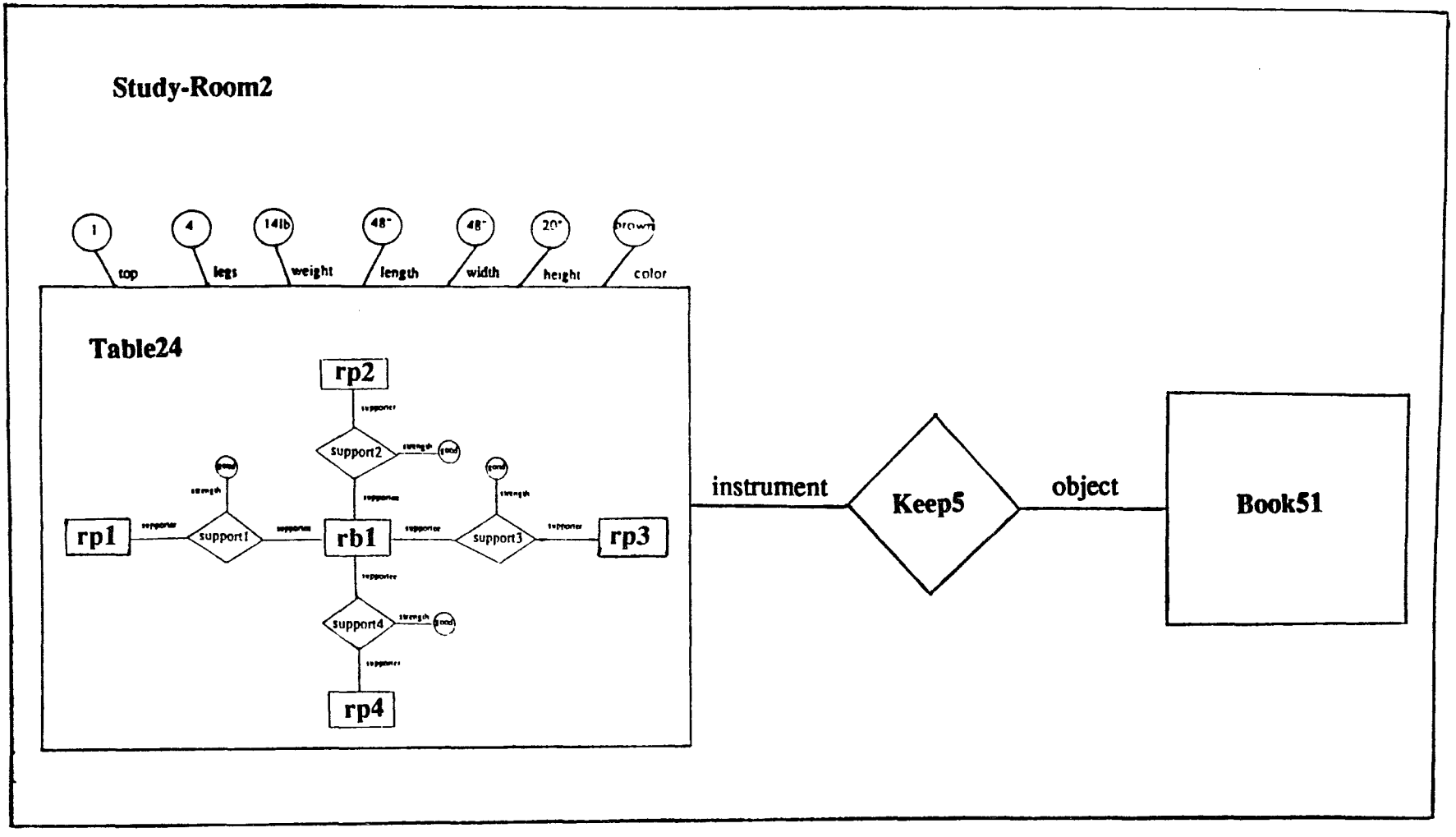


Figure 2.12. An instance ER-diagram representing external and internal descriptions of the entity 'table24'

The general description of a relationship may be summarized as the following:

Relationship:

o-enties	entities
roles	entities
attributes	values

2.6.3. Abstractions

Humans normally tend to group similar things together to form meaningful abstractions. We noted earlier that while perceiving things, we may focus our attention on the parts themselves, or the organization of the parts, or the wholes that are formed. Since entities represent both parts and wholes, and relationships represent the organization of the parts in a whole, we may group similar entities into entity sets and similar relationships into relationship sets. For example, the entity set 'table' may be used to group specific entities such as 'table1', 'table2' ..., and 'table24'. Similarly, the relationship set 'support' may be used to group specific relationships such as 'support1', 'support2', 'support3', and 'support4'.

The generic ER-diagram in figure 2.13 shows the abstraction of instance ER-diagrams such as figure 2.12. *Note that while we represent entities, relationships, and values in an instance ER-diagram by single rectangular boxes, diamonds and circles respectively, entity sets, relationship sets, and value set in a generic ER-diagram are represented by double rectangular boxes, diamonds, and circles respectively.*

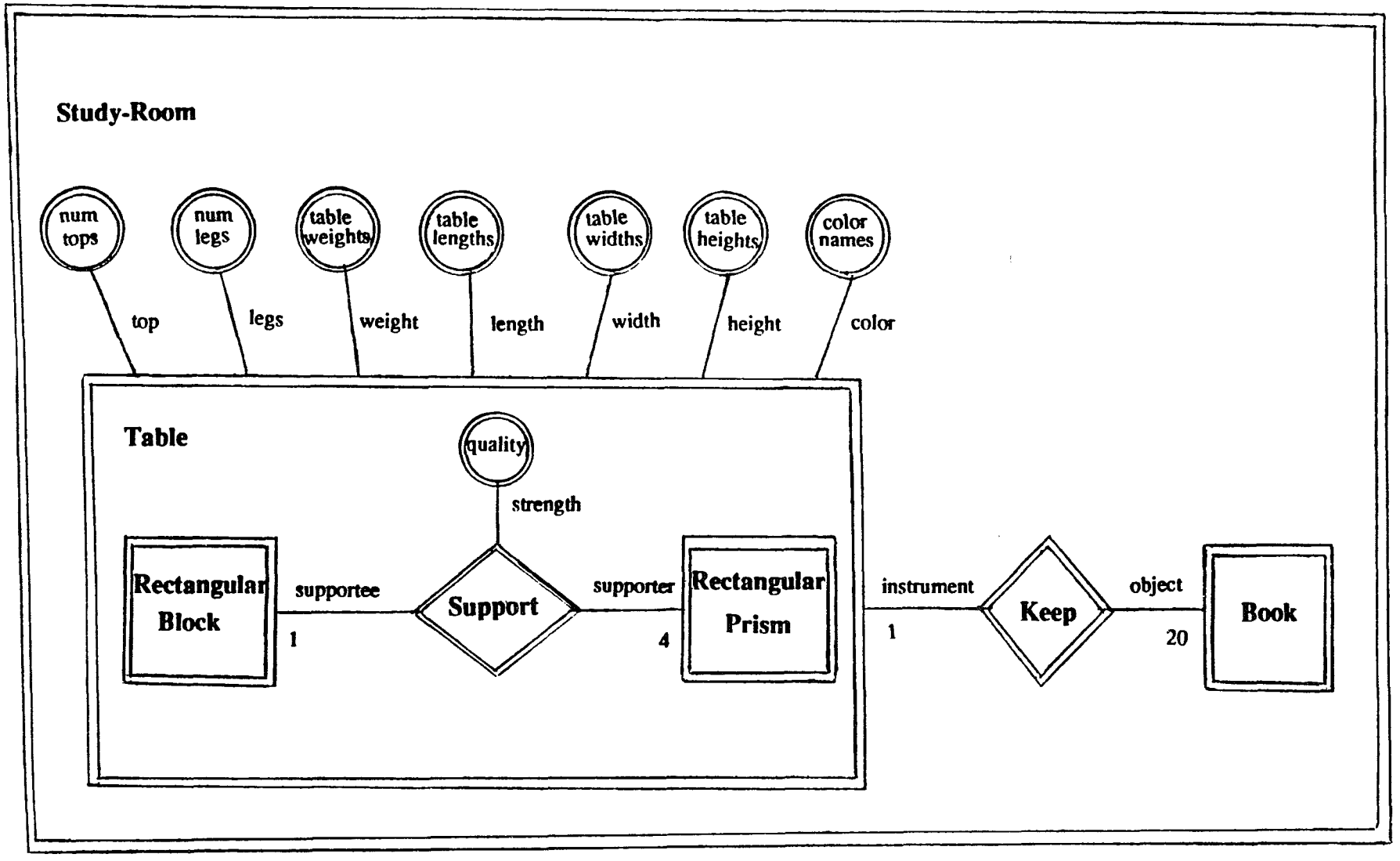


Figure 2.13. A generic ER-Diagram for external and internal descriptions of a table

2.6.3.1. Entity Sets

Entity sets in Entity-Relationship approach represent groups of similar entities. The entities in an entity set have a common description. Instead of describing a specific entity, an entity set provides a general description of all its member entities. An entity set description includes both external and internal description of its member entities.

The external description of an entity set deals with (a) the attributes of its member entities, (b) the roles played by its member entities, and (c) the w-entities representing the entity sets containing the entities in which its member entities may become the parts.

For each of its attributes, an entity set is associated with a value set such that each entity in the entity set assumes a specific value(s) from the value set for the attribute under consideration. Consider, for example, the attribute 'length' of the entity set 'table' associated with the value set 'table-lengths' in figure 2.13. Therefore, the length of any specific table entity will have a numerical value drawn from the value set 'table-lengths'. Value sets used to represent groups of values are discussed in section 2.6.3.2.

Each of the roles of an entity set associate it with a relationship set such that the entities in the entity set may perform the function specified by the role in one or more relationships in the relationship set. Consider the table playing the 'instrument' role in the relationship set 'keep'. This means that any specific table may be used as an instrument in 'keeping' different things. Relationship sets used to group similar relationships together are discussed in section 2.6.3.3.

The w-entities of an entity set associate it with other entity sets such that the entities in the entity set may become part of the entities in the associated entity sets. For example, different table entities may be used as a part of different study-room entities. Therefore, the entity set 'table' has the entity set 'study-room' as one of its w-entities.

The internal description of an entity set is concerned with (a) the entity sets containing entities that are part of the entities in the entity set, and (b) the relationship sets containing relationships that provide organization to the component entities of the entities in the entity set.

The p-entities in the internal description of an entity set associate it with the entity sets containing entities that are part of the entities in the entity set. The entity set 'table', for example, has the entity sets 'rectangular prisms' and 'rectangular blocks' as its p-entities.

The o-relationships in the internal description of an entity set associate it with the relationship sets containing the relationships that provide organization to the component entities of the entities in the entity set. For example, the entity set 'table' has the relationship set 'support' in its o-relationships containing the relationships that organize the parts of a table.

The following is the general description of an entity set:

Entity Set:

Type of Description	*Type of Domain*
external:	
attributes	value sets
roles	relationship sets

	w-entities	entity sets
internal:		
	p-entities	entity sets
	o-relationships	relationship sets

The description of an entity set may be assumed to be the criteria that must be met by an entity to become one of its members. The general description of the entity set 'table', for example, specifies the criteria that must be met by an entity to become a table.

Consider what can be mandatory or optional in the above description of an entity set. First, consider the external description of an entity set. As discussed in section 2.5.1, an entity can exist in isolation or as part of another entity. Therefore, it is not mandatory for every entity in the entity set to participate in the w-entities mentioned in the description of the entity set. The question of an entity performing any role comes only when it participates in the making of another whole. Since the participation of an entity in another whole is optional, the roles of an entity are also optional. This leaves us with the attributes of an entity. The attributes of an entity always appear with it whether it exists in isolation or it becomes a part of another entity.

Now consider the internal description of an entity. When the entity set describes atomic entities, it does not have any internal description because the entities in it are not made up of any other parts. However, if the entity set describes complex entities that are made up of other entities, then the entity set will have an internal description. Since a whole cannot be formed in the absence of its parts and the relationships that

provide organization to its parts, the p-entities and the o-relationships in the description of an entity set are treated as mandatory.

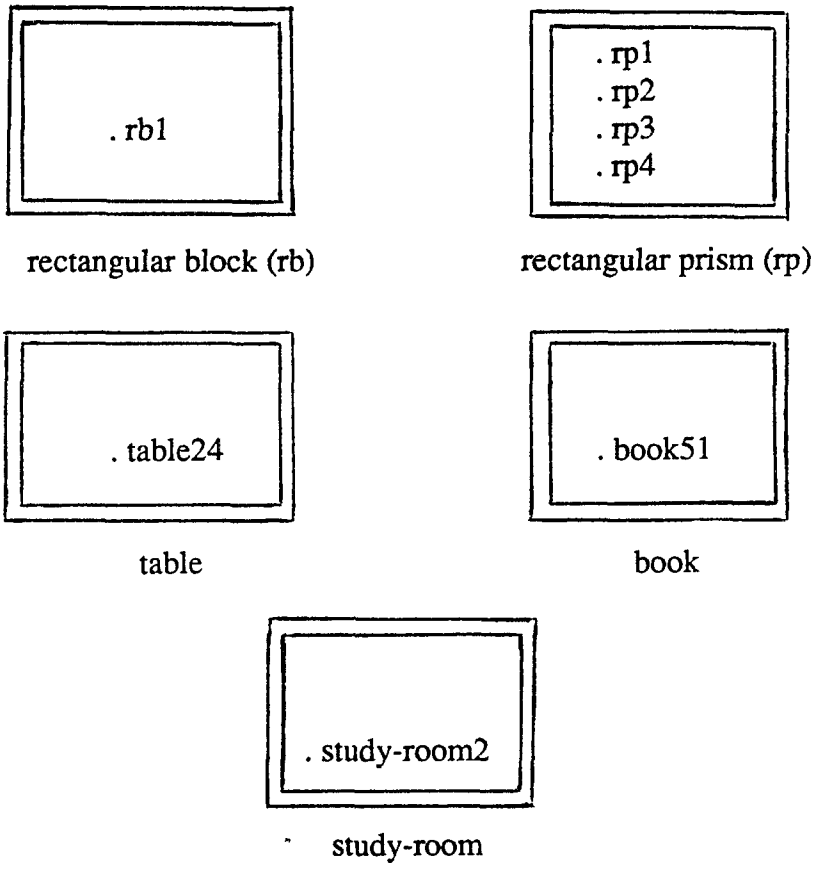


Figure 2.14. Entity set membership diagrams

In order to make the relationship between the generic diagrams (figure 2.13) and the instance diagrams (figure 2.12) explicit, we introduce set membership diagrams. An entity set membership diagram explicitly represents the entities that are the members of a given entity set. Figure 2.14 shows the entities that are the members of the entity sets in the generic ER-diagram represented by the figure 2.13.

2.6.3.2. Relationship Sets

We have seen that relationships play the central role in organizing a group of entities in forming new entities. Relationship sets in Entity-Relationship approach represent groups of similar relationships. Consider the four relationships 'support1', 'support2', 'support3', and 'support4' in figure 2.12 which may be grouped together into the relationship set 'support' as shown in figure 2.13.

The description of a relationship set includes (a) the entity sets containing the entities formed by the organization provided by its relationships, (b) the entity sets containing the entities that play specific roles in its relationships, and (c) the value sets containing the values that the attributes of its relationships assume.

The o-entities of a relationship set associate it with entity sets containing the entities formed by the organization provided by the relationships in the relationship set. For example, the o-entities of the relationship set 'support' in figure 2.13 includes the entity set 'table' indicating that the tables are one of the things that the 'support' relationships can help form.

Each role in a relationship set associates it with an entity set such that this particular role in each relationship in the relationship set is played by an entity from the corresponding entity set. In figure 2.13, the 'supporter' role associates the relationship set 'support' with the entity set 'rectangular prism'.

For each of its attributes, a relationship set is associated with a value set such that the attribute of the relationships in the relationship set under consideration assumes values from the associated value set. The attribute 'strength' in figure 2.13 associates the relationship set 'support' with the value set 'quality'.

The general description of a relationship set is summarized as the following:

Relationship Set:

Type of Description	*Type of Domain*
o-entities	entity sets
roles	entity sets
attributes	value sets

The description of a relationship set may be assumed to be the criteria that a relationship must satisfy to become a member of the relationship set. For example, the description of relationship 'support1' satisfies the description of the relationship set 'support'.

The description of a relationship may also include cardinality restrictions, if any, on its o-entities, roles, and attributes. Figure 2.13 shows, for example, the cardinality of 'object' role in the relationship set 'keep' as 20 indicating that up to 20 books can participate in the 'object' role of a single instance of 'keep' relationship.

Which of the above descriptions of a relationship set are mandatory and optional ? Since relationships are used to organize the formation of entities, their o-entities are mandatory. In a relationship, it is possible for some of its roles to be mandatory while others being optional. Consider the relationship 'break'. While it has the object role as mandatory, its other roles such as instrument and agent are optional. A relationship may or may not have attributes. They are introduced only when the relationships need to describe themselves.

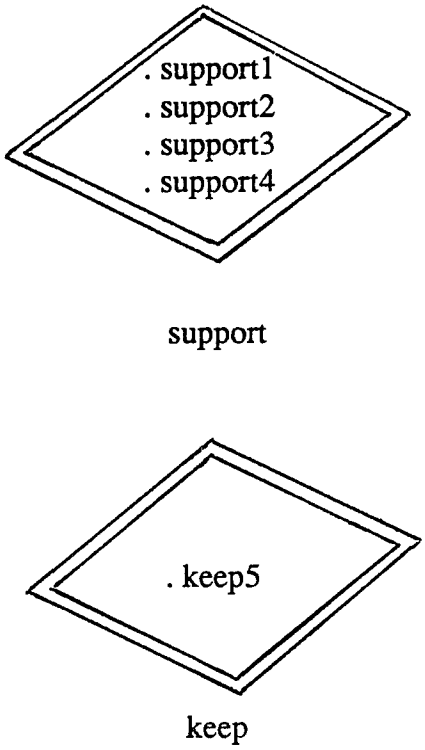


Figure 2.15. Relationship set membership diagrams

We introduce relationship set membership diagrams to indicate which of the relationships in a instance diagram (figure 2.12) are the members in a relationship set of a generic diagram (figure 2.13). A relationship set membership diagram explicitly represents the relationships that are the members of a given relationship set. Figure 2.15 shows the relationships that are the members of the relationship sets in the generic ER-diagram represented by the figure 2.13.

2.6.3.3. Value Sets

Value sets represent groups of values. A value set is formed by grouping the values that an attribute of an entity set or a relationship set may assume.

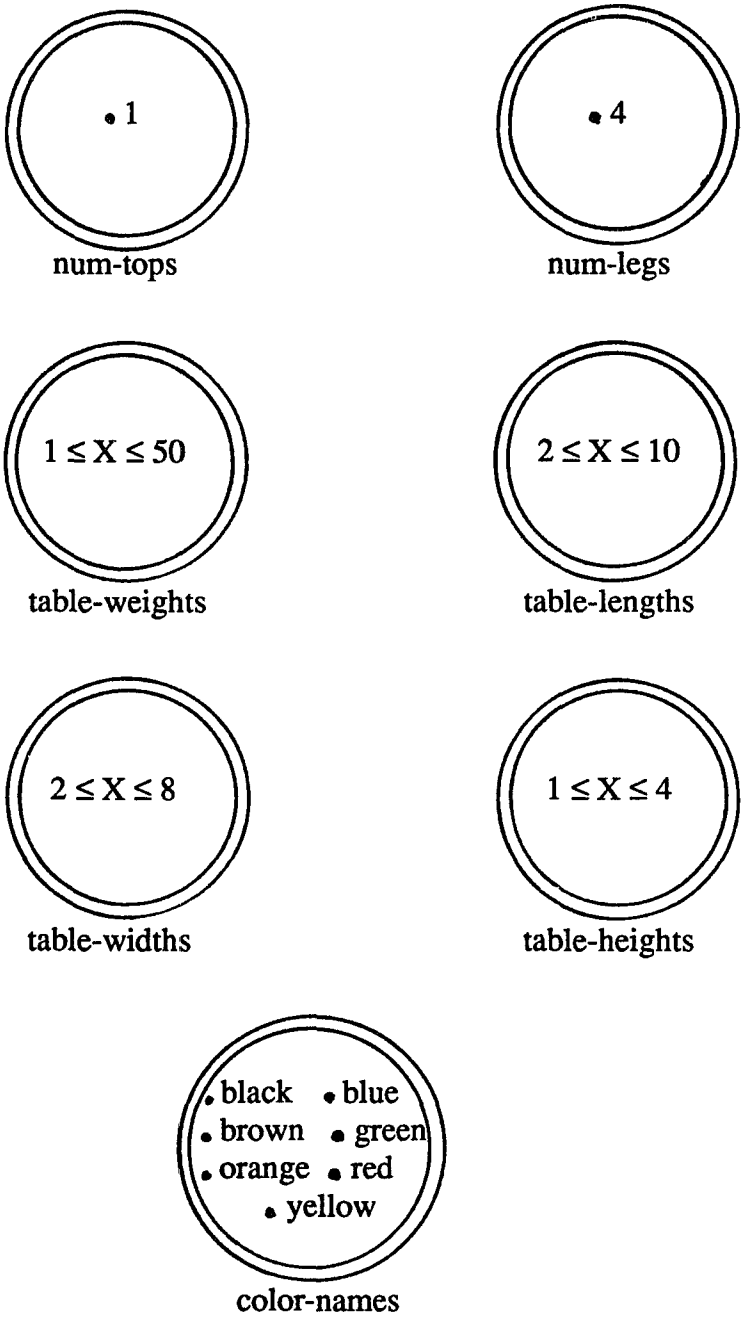


Figure 2.16. Value set membership diagrams

The set membership diagram that links the values in a instance diagram with a value set in a generic diagram is called the value set membership diagram. Figure 2.16 shows the values that are the members of the value sets in the generic ER-diagram

represented by the figure 2.13.

The members of a value set may be specified by a range of values or by enumeration. When a value set is specified as a range, a value qualifies to become a member of the value set only if it falls within the specified range of values. This form of specification is particularly suitable for quantitative values. The value set 'table-weights' associated with the attribute 'weight' of the entity set 'table' may be specified as a range of values, say, between 1 lb and 50 lb. Figure 2.16 shows the values sets: 'table-weights', 'table-lengths', 'table-widths', and 'table-heights' of the generic diagram in figure 2.13 being specified by a range.

Sometimes the members of a value set are specified by enumerating each one of them. For example, the value set 'color-names' (figure 2.16) associated with the attribute 'color' of the entity set 'table' may be enumerated as black, blue, brown, green, orange, red, and yellow. In addition to qualitative values such as colors, quantitative values may also be enumerated as in the case of the value sets 'num-tops' and 'num-legs' in figure 2.16. The members of the value sets in the generic ER-diagram, figure 2.13, are shown in figure 2.16.

2.6.4. Naming and Identification

In our notation, we have been dealing with both instances and their abstractions. Naming is a device we use to refer to them. We use generic names to represent abstractions. For example, the generic name 'table' refers to the group of specific instances of tables, each of which is referred by an individual name such as 'table1' and 'table2'. In our scheme, we use unique names for both instances and their abstrac-

tions. Therefore, instances representing entities and relationships, and abstractions representing entity sets and relationship sets have unique names.

Normally, a thing is identified by finding its name from its description. The task of identification becomes easier only if the items to be identified have a unique description. However, if there are more than one item with the same description, it is not possible to identify the given item uniquely. Consider two red balls 'r1' and 'r2' having identical description. If the description of an item that matches the description of the two red balls is given, we cannot determine which one of the two it refers to. It is equally probable that the given item may be any one of the two red balls.

On the other hand, it is always possible to find the description of an item from its name because items have unique names in our notation. Note that the names of items in our scheme correspond to ID#'s assigned in database design such as employee# and social-security#. Therefore, the names of entities, relationships, entity sets, and relationship sets allow us to find their descriptions uniquely.

Now let us consider the task of finding whether an item belongs to a group of items or not. Consider the problem of determining whether a given entity belongs to a given entity set or not. The criteria to determine whether an entity is a member of the given entity set is provided by the description of the entity set. If the description of the entity satisfies the criteria specified by the entity set description, then it qualifies to become a member of the entity set. The descriptions of entities and entity sets were discussed in sections 2.6.1 and 2.6.3.1. Similarly, whether a given relationship belongs to a given relationship set can be determined from their descriptions.

2.6.5. Forming Hierarchies

Entity sets, relationship sets, and value sets may be organized into hierarchies from their descriptions. For example, given the description of the entity sets, 'table' and 'chair', a more general entity set called 'artifact' may be formed by abstracting out their common descriptions. The advantage of abstracting descriptions in a hierarchical form is that it allows compact representation of classes of things by permitting inheritance of properties from more general classes to specific classes [EtR83, Fox79, Tou86, Woo83].

In most representations, different types of descriptions are treated uniformly as properties. However, we have seen in this chapter that different types of descriptions must be treated separately such as wholes, parts, properties, roles, and organizing relationships. Therefore, a class of things must be allowed to inherit properties as well as other descriptions from their more general classes.

2.6.6. Consequences to Inheritance

The observation that *wholes* having properties different from their *parts* has important consequences to property inheritance. It points out the inappropriateness of property inheritance from the parts to the whole in a *hasa* hierarchy. Actually, what a whole inherits is the parts of its parts than the properties of its parts. Properties of the whole depend on the configuration of its parts. What emerges as a whole from the parts need not necessarily have the same properties as its parts and may even have new properties. Therefore, a *hasa* hierarchy must take organizational aspects of the parts of a whole into consideration to determine what can be inherited from the parts

to the whole.

Unlike a *hasa* hierarchy, we find that the property inheritance takes place in a *isa* hierarchy. Instead of dealing with parts and wholes relationship, a *isa* hierarchy deals with abstractions among wholes. For example, the abstraction of the things (wholes) such as an 'elephant' and a 'horse' as a 'mammal' in a *isa* hierarchy does not treat the elephant and the horse as the two parts of the mammal. Instead, mammal is an abstraction used to represent both the elephant and the horse by factoring out their common descriptions. What is inherited in a *isa* hierarchy is both properties and parts form general classes to specific classes.

Therefore, we find that a *hasa* hierarchy should permit only parts to be inherited instead of properties from parts to wholes. On the other hand, a *isa* hierarchy should permit the inheritance of properties as well as other descriptions.

CHAPTER III

A Clausal Form Implementation Of An ER-Knowledge Base System

3.1. Introduction

Chapter II provided a basis for using Entity-Relationship approach for the analysis and the specification of a knowledge base system. In this chapter, we develop a framework to guide the implementation of a knowledge base as a first-order logic system or as a production system from the specifications provided by Entity-Relationship approach. Translation rules for the conversion of ER-diagrams into clausal form are provided by mapping appropriate symbolic data structures. We have chosen clausal form because it provides a common ground for the implementation of both logic programming systems and production systems.

3.2. Clausal Form and Predicate Calculus

Predicate calculus can be expressed in more than one form [Bun83, ChL73, Kow79a, Llo84, WOL84]. Clausal form of logic has gained a significant attention in recent years because it lead to the concept of using logic as a programming language [Dav85, GeG85, KaC87, Kow74]. Prolog is a programming language based on clausal form of logic. In this chapter, we will examine clausal form and then show how the ER-diagrams can be translated into clausal form. Since there are standard algorithms available for the conversion of clausal form into other forms, the conversion of ER-diagrams into any form of logic can be obtained

mechanically from its clausal form representation.

3.3. Clausal Form and Production Systems

Clausal form provides a natural representation for production systems because clauses can be classified into either rules or facts. Although both deductive systems and production systems are based on clausal form representation, they differ in their styles of reasoning. Deductive systems based on logic programming are goal-oriented and employ backward chaining among the clauses. Production systems are based on 'recognize situation and act' paradigm in which forward chaining of clauses is employed [BaF81, For81, Hay85, KCP87, New73, Wat86].

3.4. Clausal Form

Any form of logic generates well-formed sentences. In clausal form, sentences are generated as a collection of clauses. The following is a formal definition of clausal form of logic [Bun83, Kow79a, Kow79b]:

> A *sentence* is a collection of clauses.
> A *clause* is an expression of the form
>
> $B_1, ..., B_m \leftarrow A_1, ..., A_n$ m,n ≥ 0,
>
> where $A_1, ..., A_n$ are called the *conditions* of the clause and $B_1, ..., B_m$ are called the *conclusions*. Both conditions and conclusions are expressions of the form
>
> $P(t_1, ..., t_k)$
>
> called *atoms*, where P is a k-argument predicate symbol and $t_1, ..., t_k$ are *terms*. Terms are either constants, variables, or *functional terms* which are expressions of the form
>
> $f(t_1, ..., t_l)$
>
> where f is an l-argument function symbol and $t_1, ..., t_l$ are terms.

In the following sections, we will examine the clausal form in detail to devise translation rules for the implementation of the knowledge base from ER-diagrams.

3.4.1. Terms

According to the above definition of clausal form, terms represent either constants, variables, or functions. Let us consider what each of them stand for in Entity-Relationship approach.

3.4.1.1. Constants

Constants in clausal form indicate particular individuals. Therefore, they allow us to represent individual entities, relationships, attributes, values, and roles in Entity-Relationship approach. For example, particular entities such as 'table24', and 'rb1' are represented by constants. Specific relationships such as 'keep5', and 'support1' become constants. Similarly, specific attributes, values, and roles are represented by constants.

Constants are represented in our notation by names that start with a lower case letter as in 'table24', and 'keep5'. Since all constants are treated uniformly in clausal form, we introduce predicates in section 3.4.2.2.1 that distinguish the different types of individuals in Entity-Relationship approach.

3.4.1.2. Variables

A variable is a term that represents an arbitrary individual. Hence, variables are useful in representing arbitrary entities, relationships, roles, attributes, and values in Entity-Relationship approach. As an illustration, the variable 'X' in the predicate

'entity(X)' stands for an arbitrary entity.

In our notation, we indicate variables with the names that start with an upper case letter as in 'X'.

3.4.1.3. Functions

Functional terms represent a relation between two objects. Consider 'color' as an attribute, which may be represented as a function between an entity and a value, say 'table24' and 'brown':

color(table24) = brown.

Although functions are a useful notation, we avoid them by adopting an alternative predicate notation due to the reasons cited in section 3.4.2.1.

3.4.2. Predicates

Simple assertions in clausal form are represented by atomic formulas. Each atomic formula in clausal form is represented by a predicate. For example, the predicate

$P(t_1, ..., t_n)$

represents an atomic formula, where P is an n-place predicate symbol and $t_1, ..., t_n$ are the terms. We interpret this atomic formula as an assertion of the relation called P among the individuals, $t_1, ..., t_n$. As an example, the predicate

color(table24,brown)

represents the simple assertion that 'the color of the table24 is brown'.

3.4.2.1. Functions or Predicates ?

Given an arbitrary relation, r, between arbitrary objects, x and y, we can represent it with a function and equality, i.e., r(x) = y or with a predicate, i.e., r(x,y). Which of the two forms, the functions and the predicates, is a more preferable notation ?

First, the predicate notation is a more natural form of representation for production systems than the functional notation. Second, functions are avoided in practical implementations due to the difficulties introduced by the equality [Bun83, Rei78, WRC65]. Therefore, we adopt a predicate notation and replace functional terms. Thus, the function

color(table24) = brown

is replaced by the predicate

color(table24,brown).

In the absence of functional terms, the predicates in our representation will have only constants and variables as their arguments.

3.4.2.2. A Predicate Notation

As noted earlier, a predicate is a syntactic representation of a general relationship among n-individuals. Since Entity-Relationship approach emphasizes on the semantic distinction among different types of individuals and their associations, we must devise an explicit predicate notation. In the following subsections, we propose a predicate notation for the representation of ER-diagrams.

3.4.2.2.1. Types of Individuals

We described in section 3.4.1.1 that constants in clausal form represent individuals. While all individuals in clausal form are treated alike, they are distinguished in Entity-Relationship approach into different types. Constants that represent entities, relationships, attributes, values, and roles must be distinguished. Therefore, we introduce unary predicates in which the predicate argument indicates the name of the individual, and the predicate name indicates the type to which the individual belongs to. An individual entity such as 'table24', for example, is represented by the predicate:

entity(table24).

Individuals in Entity-Relationship approach may be classified into one of the six categories: entities, relationships, entity attributes, relationship attributes, values, and roles. Therefore, we introduced six unary predicates 'entity(x)', 'relationship(x)', 'entity-attribute(x)', 'relationship-attribute(x)', 'value(x)', and 'role(x)' to explicitly indicate the category to which a given individual 'x' belongs to.

3.4.2.2.2. Types of Sets

Sets in Entity-Relationship approach may be classified into one of the three types: entity sets, relationship sets, and value sets. We need predicates that distinguish these three different types of sets. Therefore, we introduce three unary predicates 'entity-set(x)', 'relationship-set(x)', and 'value-set(x)' to explicitly state what a given set stands for. If 'table' is the name of an entity set, then it is indicated by the predicate 'entity-set(table)'.

3.4.2.2.3. Individuals in a Set

Sets often have members. They are usually specified as simple assertions. We have seen in the preceding section that sets themselves can be classified into different types. To assert members into each of the three different types of sets: entity sets, relationship sets, and value sets, we introduce three corresponding binary predicates 'entity-set-member(x,y)', 'relationship-set-member(x,y)', and 'value-set-member(x,y)'. Each of the three predicates must be read as 'x is a member of entity set y', 'x is a member of relationship set y' and 'x is a member of value set y' respectively.

The following asserts that 'table24' is a member of the entity set 'table', 'keep5' is a member of the relationship set 'keep', and '6' is a member of the value set 'table-weights', then they are represented as:

entity-set-member(table24,table).

relationship-set-member(keep5,keep).

value-set-member(6,table-weights).

3.4.2.2.4. Predicates for Descriptions

In order to translate the notation in section 2.6 of chapter II, we introduce predicates for the description of entities, relationships, entity sets, relationship sets, and value sets. Sections 3.4.4 to 3.4.8 describe the predicates introduced for this purpose and their meaning.

3.4.3. Clauses

If we assume that each predicate represents a simple assertion, clauses are a more general form of expressions that encompass simple assertions also. Consider our earlier definition of a clause, which is an expression of the form

$$B_1, ..., B_m \leftarrow A_1, ..., A_n \quad m,n \geq 0,$$

where $B_1, ..., B_m, A_1, ..., A_n$ are atoms. The atoms $A_1, ..., A_n$ are *conditions* of the clause and the atoms $B_1, ..., B_m$ are *conclusions* of the clause.

3.4.3.1. Types of Clauses

Based on the above definition, we can classify clauses into one of the following four types:

1. unconditional assertional clauses $B_1, ..., B_m \leftarrow$
2. conditional assertional clauses $B_1, ..., B_m \leftarrow A_1, ..., A_n$
3. goal clauses $\leftarrow A_1, ..., A_n$
4. empty clause $\leftarrow$

While the first two types of clauses are useful for the representation of a knowledge base, the last two types of clauses allow the user to query and obtain answers from the knowledge base. Unconditional assertions represent facts that are believed to be true.

In each conditional assertion, the clause has conditions and conclusions. Conditional assertions are useful in representing rules. A general conditional assertional clause indicates that if assertions $A_1, ..., A_n$ are true, then the assertions $B_1, ..., B_n$ are

true. We will propose a number of conditional clauses in the following sections that must be satisfied for the implementation of ER-diagrams in clausal form.

Goal clauses are useful in querying the knowledge base. Any query of interest to the user is formulated as a goal clause. Empty clause plays an important role in proof procedures such as resolution [Rob65] that provide answers to queries to the knowledge base.

3.4.3.2. Quantification

Note that clauses contain no quantifiers. Universal quantification is indicated by leaving variables free. Existential quantification is indicated by the introduction of new functions and constants called skolem functions and skolem constants. Since we avoid the use of functions in our representation (section 3.4.2.1), we will use only constants to indicate existential quantifiers.

3.4.4. Entities

Every entity has a name. The predicate 'entity(x)' indicates that x is the name of an entity. As an example, the entity 'table24' in figure 2.12 is represented as the following:

entity(table24).

In addition to the name, every entity has a description also. Before attempting to provide the complete description of an entity, we introduce the predicates needed to represent each type of description that an entity may have.

The description provided by each attribute of an entity is represented by the

predicate 'entity-attribute-description(x,y,z)', which is read as 'entity x has the attribute y with the value z'. For example, the 'table24' having the attribute 'weight' with a value '14' lb is represented by the following:

entity-attribute-description(table24,weight,14).

The predicate 'entity-role-description(x,y,z)' represents that entity 'x' plays the role 'y' in relationship 'z'. Hence, the 'table24' playing the 'instrument' role in relationship 'keep5' is represented as follows:

entity-role-description(table24,instrument,keep5).

To indicate that a given entity 'x' is a part of another whole 'y', we introduce the predicate 'w-entity(x,y)'. The fact that 'table24' is part of the 'study-room2' is represented by the following predicate:

w-entity(table24,study-room2).

An entity may be made up of other parts. If the entity 'x' has the entity 'y' as one of its parts then it is represented by the predicate 'p-entity(x,y)'. For example, 'table24' having 'rectangular-block1 (rb1)' as one of its parts is represented by the following:

p-entity(table24,rb1).

When an entity is formed out of other entities, it is organized by the relationships among its parts. The predicate 'o-relationship(x,y)' represents that the relationship 'y' provides organization to the entity 'x'. The relationship 'support1' providing the organization to the entity 'table24' in figure 2.12 is represented by:

o-relationship(table24,support1).

After defining the predicates needed to represent different types of entity descriptions, let us now consider the representation of the complete description of an entity. The complete description of an entity 'x' is known only if the external and the internal descriptions of the entity 'x' are known. Let the three predicates 'entity-description(x)', 'entity-external-description(x)', and 'entity-internal-description(x)' stand for complete description, external description, and internal description of the entity 'x' respectively.

Consider the external description of an entity 'x'. The external description of the entity 'x' is known only if all of its attributes, the roles it plays in other relationships, and the wholes in which it participates are known. Assume that the entity 'x' has the attributes 'a_1', 'a_2' ..., and 'a_m' with the corresponding values 'v_1', 'v_2', ..., and 'v_m'. Let the entity 'x' be playing the roles 'r_1', 'r_2', .., 'r_n' in the corresponding relationships 'rel_1', 'rel_2', ..., and 'rel_n'. Assume that the entity 'x' is a part of the wholes 'w_1', 'w_2', ..., 'w_o'. Then we can represent our earlier statement that the external description of the entity 'x' is known only if all of its attributes, roles, and w-entities are known by the following clause:

entity-external-description(x) ← entity-attribute-description(x,a_1,v_1),
entity-attribute-description(x,a_2,v_2),
...
entity-attribute-description(x,a_m,v_m),
entity-role-description(x,r_1,rel_1),
entity-role-description(x,r_2,rel_2),
...
entity-role-description(x,r_n,rel_n),
w-entity(x,w_1),
w-entity(x,w_2),

...
w-entity(x,w_o).

If any attribute has multiple values, then two or more of the above attribute names become identical. Similarly, if the entity plays the same role in more than one relationship, then two or more of the above role names become identical. The following is the external description of 'table24' in figure 2.12:

entity-external-description(table24) ←
entity-attribute-description(table24,top,1),
entity-attribute-description(table24,legs,4),
entity-attribute-description(table24,weight,14),
entity-attribute-description(table24,length,48),
entity-attribute-description(table24,width,4),
entity-attribute-description(table24,height,4),
entity-attribute-description(table24,color,brown),
entity-role-description(table24,instrument,keep5),
w-entity(table24,study-room2).

Similarly, the internal description of the entity 'x' is known only if all of its parts and their organizing relationships are known. Let the entity 'x' have the entities 'p_1', 'p_2', ..., and 'p_r' as its parts and the relationships 'o_1', 'o_2', ..., 'o_s' organizing its parts, then:

entity-internal-description(x) ← p-entity(x,p_1),
p-entity(x,p_2),
...
p-entity(x,p_r),
o-relationship(x,o_1),
o-relationship(x,o_2),
...
o-relationship(x,o_s).

The following is an example of the internal description of 'table24':

entity-internal-description(table24) ← p-entity(table24,rb1),
p-entity(table24,rp1),
p-entity(table24,rp2),

p-entity(table24,rp3),
p-entity(table24,rp4),
o-relationship(table24,support1),
o-relationship(table24,support2),
o-relationship(table24,support3),
o-relationship(table24,support4).

When both internal and external descriptions of any entity 'X' are known, then its description is known. Therefore, we have the following general clause:

entity-description(X) ← entity-external-description(X),
entity-internal-description(X).

The above clause can be used to infer the description of any specific entity such as 'table24' once its external and internal descriptions are known.

3.4.5. Relationships

Since every relationship has a name, we introduce the predicate 'relationship(x)' to indicate that the name 'x' stands for a relationship. The relationship 'support1' in figure 2.12 is represented by the following predicate:

relationship(support1).

In addition to the name, a relationship has a description also. Let us introduce the predicates that are needed to represent different types of descriptions that a relationship may have.

In a relationship, each entity participating in it plays a specific role. The predicate 'relationship-role-description(x,y,z)' is used to represent that the role 'y' in the relationship 'x' is played by the entity 'z'. The following represents that the 'supportee' role in relationship 'support1' is played by the entity 'rectangular-block (rb1)':

relationship-role-description(support1,supportee,rb1).

The entity 'y' formed out of the organization provided by the relationship 'x' is represented by the predicate 'o-entity(x,y). The following represents that relationship 'support1' helps form the entity 'table24':

o-entity(support1,table24).

A relationship can have its own attributes to describe itself. Therefore, we introduce the predicate 'relationship-attribute-description(x,y,z)' read as 'the relationship x has the attribute y with the value z'. The relationship 'support1' having the attribute 'strength' in figure 2.12 is represented by:

relationship-attribute-description(support1,strength,good).

Consider the complete description of any relationship 'x'. Assume that the relationship 'x' helps organize the formation of the entities 'w_1', 'w_2', ..., 'w_t'. Let the roles 'r_1', 'r_2' ..., 'r_u' in the relationship 'x' be performed by the entities 'p_1', 'p_2', ..., 'p_u' respectively. Assume that the relationship 'x' also has the attributes 'a_1' 'a_2', ... 'a_v' having the values 'v_1', 'v_2', ..., 'v_v' respectively. The fact that the complete description of the relationship 'x' is known only when all the entities it helps form, the roles played by the entities participating in it, and its own attributes are known can be expressed by the following clause:

relationship-description(x) ← o-entity(x,w_1),
o-entity(x,w_2),
...
o-entity(x,w_t),
relationship-role-description(x,r_1,p_1),
relationship-role-description(x,r_2,p_2),
...
relationship-role-description(x,r_u,p_u),

relationship-attribute-description(x,a_1,v_1),
relationship-attribute-description(x,a_2,v_2),
...
relationship-attribute-description(x,a_v,v_v).

The following provides the description of the relationship 'support1' in figure 2.12:

relationship-description(support1) ←
o-entity(support1,table1),
relationship-role-description(support1,supporter,rp1),
relationship-role-description(support1,supportee,rb1),
relationship-attribute-description(support1,strength,good).

We may note that some facts about a relationship are also relevant to an entity related to it. For example, the predicate 'relationship-role-description(x,y,z)' of relationship 'x' represents the same information as the predicate 'entity-role-description(z,y,x)' of entity 'z'. If one of them is known the other can be inferred from the relation between them expressed by the following two general clauses:

entity-role-description(X,Y,Z) ← relationship-role-description(Z,Y,X).

relationship-role-description(X,Y,Z) ← entity-role-description(Z,Y,X).

The predicate 'o-entity(x,y)' of relationship 'x' represents the same information as the predicate 'o-relationship(y,x) of entity 'y'. Therefore, they can be related by the following two general clauses:

o-entity(X,Y) ← o-relationship(Y,X).

o-relationship(X,Y) ← o-entity(Y,X).

The predicate 'w-entity(x,y)' of entity 'x' represents the same information as the predicate 'p-entity(y,x)' of entity 'y'. Therefore, these two predicates can be related by the following two clauses:

w-entity(X,Y) ← p-entity(Y,X).

p-entity(X,Y) ← w-entity(Y,X).

In order to represent facts only once and infer the related facts, we will adopt the following strategy: create only the facts of the type 'entity-role-description(x,y,z)', 'w-entity(x,y)', 'o-relationship(x,y)' and infer the facts of the type 'relationship-role-description(z,y,x)', 'p-entity(y,x)', and 'o-entity(y,x)' from the following general clauses:

relationship-role-description(Z,Y,X) ← entity-role-description(X,Y,Z).

o-relationship(X,Y) ← o-entity(Y,X).

p-entity(X,Y) ← w-entity(Y,X).

3.4.6. Entity Sets

Entity sets have names. The predicate 'entity-set(x)' indicates that the name 'x' stands for an entity set. The entity set 'table' in figure 2.13, for example, is represented as follows:

entity-set(table).

Entity sets contain individual entities as their members. Each member 'x' in the entity set 'y' is indicated by the predicate 'entity-set-member(x,y)'. This predicate is useful in translating an entity set membership diagram, such as figure 2.14. The following indicates that 'table24' is a member of the entity set 'table':

entity-set-member(table24,table).

The description of any entity set 'x' specifies the criteria that must be met by any

entity 'y' to become its member. As mentioned in section 2.6.3.1, the description of an entity set consists of mandatory and optional descriptions. While the mandatory descriptions of an entity set must be satisfied by all of its members, the optional descriptions need not necessarily be satisfied by all of its members. In the external description of an entity set, only its attributes are mandatory while its associated roles and w-entities are optional. The internal description of an entity set includes mandatory parts and their relationships. Assume that entity set 'y' having attributes 'a_1', 'a_2', ... 'a_p' associated with value sets '$vset_1$', '$vset_2$', ..., '$vset_p$', entity sets '$eset_1$', '$eset_2$'. ..., '$eset_q$' as its p-entities, and relationship sets '$rset_1$', '$rset_2$', ..., '$rset_r$' as its o-relationships. Then any entity 'X' which is a member of the entity set 'y' must satisfy the following clause:

entity-set-member(X,y) ←
    entity-attribute-description(X,a_1,V_1),
        value-set-member(V_1,$vset_1$),
    entity-attribute-description(X,a_2,V_2),
        value-set-member(V_2,$vset_2$),
        ... ...
    entity-attribute-description(X,a_p,V_p),
        value-set-member(V_p,$vset_p$),
    p-entity(X,E_1),
        entity-set-member(E_1,$eset_1$),
    p-entity(X,E_2),
        entity-set-member(E_2,$eset_2$),
        ... ...
    p-entity(X,E_q),
        entity-set-member(E_q,$eset_q$),
    o-relationship(X,R_1),
        relationship-set-member(R_1,$rset_1$),
    o-relationship(X,R_2),
        relationship-set-member(R_2,$rset_2$),
        ... ...
    o-relationship(X,R_r),
        relationship-set-member(R_r,$rset_r$).

The description of the entity set 'table' in figure 2.13 is given below:

entity-set-member(X,table) ←
entity-attribute-description(X,top,V_1),
value-set-member(V_1,num-tops),
entity-attribute-description(X,legs,V_2),
value-set-member(V_2,num-legs),
entity-attribute-description(X,weight,V_3),
value-set-member(V_3,table-weights),
entity-attribute-description(X,length,V_4),
value-set-member(V_4,table-lengths),
entity-attribute-description(X,width,V_5),
value-set-member(V_5,table-widths),
entity-attribute-description(X,height,V_6),
value-set-member(V_6,table-heights),
entity-attribute-description(X,color,V_7),
value-set-member(V_7,color-names),
p-entity(X,E_1),
entity-set-member(E_1,rectangular-prism),
p-entity(X,E_2),
entity-set-member(E_2,rectangular-prism),
p-entity(X,E_3),
entity-set-member(E_3,rectangular-prism),
p-entity(X,E_4),
entity-set-member(E_4,rectangular-prism),
p-entity(X,E_5),
entity-set-member(E_5,rectangular-prism),
o-relationship(X,R_1),
relationship-set-member(R_1,support),
o-relationship(X,R_2),
relationship-set-member(R_2,support),
o-relationship(X,R_3),
relationship-set-member(R_3,support),
o-relationship(X,R_4),
relationship-set-member(R_4,support).

3.4.7. Relationship Sets

Every relationship set has a name. The predicate 'relationship-set(x)' indicates that 'x' is the name of an entity set. The relationship set 'support' in figure 2.13 is represented by:

relationship-set(support).

The members of a relationship is indicated by the predicate 'relationship-set-member(x,y)'. For example, the relationship set 'support' having the relationship 'support1' as its member (figure 2.15) is represented as follows:

relationship-set-member(support1,support).

The description of a relationship set specifies the criteria that a relationship must satisfy to become its member. The discussion in section 2.6.2 revealed that some roles, attributes, and o-entities in the description of a relationship set are mandatory. Assume that the relationship set 'y' having the entity sets '$eset_1$', '$eset_2$', ..., '$eset_u$' as its mandatory o-entities, the entity sets 'est_1', 'est_2', ..., 'est_v' associated with its roles 'r_1', 'r_2', ..., 'r_v', and the value sets '$vset_1$', '$vset_2$', ..., '$vset_w$' associated with the its attributes 'a_1', 'a_2', ..., 'a_w'. Then, every relationship 'X' in the relationship set 'y' must satisfy the following clause:

relationship-set-member(X,y) ←
o-entity(X,Z_1),
entity-set-member(Z_1,$eset_1$),
o-entity(X,Z_2),
entity-set-member(Z_1,$eset_2$),
... ...
o-entity(X,Z_u),
entity-set-member(Z_u,$eset_u$),
relationship-role-description(X,r_1,E_1),
entity-set-member(E_1,$eset_1$),
relationship-role-description(X,r_2,E_2),
entity-set-member(E_2,$eset_2$),
... ...
relationship-role-description(X,r_v,E_v),
entity-set-member(E_n,$eset_v$),
relationship-attribute-description(X,a_1,V_1),
value-set-member(V_1,$vset_1$),
relationship-attribute-description(X,a_2,V_2),

value-set-member(V_2,$vset_2$),
... ...
relationship-attribute-description(X,a_w,V_w),
value-set-member(V_w,$vset_w$).

The following is the description of the relationship set 'support' (figure 2.13) that every relationship in it must satisfy:

relationship-set-member(X,support) ←
o-entity(X,Z_1),
entity-set-member(Z_1,table),
relationship-role-description(X,supportee,E_1),
entity-set-member(E_1,rectangular-block),
relationship-role-description(X,supporter,E_2),
entity-set-member(E_2,rectangular-prism),
relationship-attribute-description(X,strength,V_1),
value-set-member(V_1,quality).

3.4.8. Value Sets

Value sets group values associated with the attributes of entities in a entity set or relationships in a relationship set together.

Each value set has a name indicated by the predicate 'value-set(x)'. The name of the value set 'table-weights' is represented as follows:

value-set(table-weights).

As noted in section 2.6.3.3, value sets may be specified by enumeration or by a range. When values in a value set are specified by enumeration, each value 'x' in the value set 'y' is represented explicitly by the predicate 'value-set-member(x,y)'. Therefore, the values in the value set 'color-names': black, blue, brown, green, orange, red, and yellow in figure 2.16 are specified the by the following:

value-set-member(black,color-names).

value-set-member(blue,color-names).

value-set-member(brown,color-names).

value-set-member(green,color-names).

value-set-member(orange,color-names).

value-set-member(red,color-names).

value-set-member(yellow,color-names).

A value set may sometimes be specified by a range of values. Consider the value set 'y' restricting its values between 'a' and 'b'. Then, any value 'X' in the value set 'y' satisfies the following clause:

value-set-member(X,y) $\leftarrow$ $X \geq a,\ X \leq b.$

For example, the value set 'table-weights', in figure 2.16, associated with the attribute 'weight' of a table is restricted to the range between 1 and 50 lbs, then it is represented by the following:

value-set-member(X,table-weights) $\leftarrow$ $X \geq 1, X \leq 50.$

Note that value sets were included in the description of entity sets and relationship sets in sections 3.4.6 and 3.4.7, because they are associated with attributes of entity sets or relationship sets.

3.5. Translation Rules for the Conversion of ER-diagrams:

The following provides the summary of the translation rules proposed in this chapter for the conversion of ER-diagrams into clausal form:

Rules for classifying different types of constants:

1. For each entity 'x', create the predicate 'entity(x)'
2. For each relationship 'x', create the predicate 'relationship(x)'.
3. For each role 'x', create the predicate 'role(x)'.
4. For each entity attribute 'x', create the predicate 'entity-attribute(x)'.
5. For each relationship attribute 'x', create the predicate 'relationship-attribute(x)'.
6. For each value 'x', create the predicate 'value(x)'.

Rules for classifying different types of sets:

7. For each entity set 'x', create the predicate 'entity-set(x)'.
8. For each relationship set 'x', create the predicate 'relationship-set(x)'.
9. For each value set 'x', create the predicate 'value-set(x)'.

Rules for asserting members of different types of sets:

10. For each member 'x' of the entity set 'y', create the predicate 'entity-set-member(x,y)'.
11. For each member 'x' of the relationship set 'y', create the predicate 'relationship-set-member(x,y)'.

12. For each member 'x' of the enumerated value set 'y', create the predicate 'value-set-member(x,y)'.

13. For any member 'X' in the value set 'y' specified by the range of values between 'a' and 'b', create the following clause:

    value-set-member(X,y) $\leftarrow$ $X \geq a$, $X \leq b$.

Rules for the description of entities:

14. For each value 'z' of the attribute 'y' of the entity 'x', create the predicate 'entity-attribute-description(x,y,z)'.

15. For each relationship 'z' in which the role 'y' is played by the entity 'x', create the predicate 'entity-role-description(x,y,z)'.

16. For each whole 'y' in which the entity 'x' participates, create 'w-entity(x,y)'.

17. For each part 'y' of the entity 'x', infer the predicate 'p-entity(x,y)' from the following clause linking rules 16 and 17:

    p-entity(X,Y) $\leftarrow$ w-entity(Y,X).

18. For each relationship 'y' that provides organization to the parts of the entity 'x', create the predicate 'o-relationship(x,y)'.

19. For any entity 'X', create the following clause to obtain its description:

    entity-description(X) $\leftarrow$ entity-external-description(X),
    entity-internal-description(X).

20. Given an entity 'x' with attributes 'a_1', 'a_2', ..., 'a_m' having values 'v_1', 'v_2', ..., 'v_m', performing roles 'r_1', 'r_2', ... 'r_n' in relationships 'rel_1', 'rel_2', ..., 'rel_n', and being a part of the entities 'w_1', 'w_2', ..., 'w_o', create the following clause to

obtain its external description:

entity-external-description(x) ← entity-attribute-description(x,a_1,v_1),
entity-attribute-description(x,a_2,v_2),
...
entity-attribute-description(x,a_m,v_m),
entity-role-description(x,r_n,rel_n),
entity-role-description(x,r_2,rel_2),
...
entity-role-description(x,r_m,rel_m),
w-entity(x,w_1),
w-entity(x,w_2),
...
w-entity(x,w_o).

21. Given an entity 'x' made up of entities 'p_1', 'p_2', ..., 'p_r' which are organized by the relationships 'o_1', 'o_2', ..., 'o_s', create the following clause to obtain its internal description:

entity-internal-description(x) ← p-entity(x,p_1),
p-entity(x,p_2),
...
p-entity(x,p_r),
o-relationship(x,o_1),
o-relationship(x,o_2),
...
o-relationship(x,o_s).

Rules for the description of relationships:

22. For each whole 'y' in which the relationship 'x' provides the organization to its parts, infer the predicate 'o-entity(X,Y)' form the following clause linking rules 18 and 22:

o-entity(X,Y) ← o-relationship(Y,X).

23. For each entity 'z' playing the role 'y' in the relationship 'x', infer the predicate 'relationship-role-description(x,y,z)' from the following clause linking the rules 15 and 20:

relationship-role-description(X,Y,Z) ← entity-role-description(Z,Y,X).

24. For each value 'z' of the attribute 'y' of the relationship 'x', create the predicate 'relationship-attribute-description(x,y,z)'.

25. Given a relationship 'x' that provides organization to the entities 'w_1', 'w_2', ..., 'w_t', its roles 'r_1', 'r_2', ..., 'r_u' being played by the entities 'p_1, 'p_2', ..., 'p_u', its attributes 'a_1', 'a_2', ..., 'a_v' having values 'v_1', 'v_2', ..., 'v_v', create the following clauses to obtain its description:

relationship-description(x) ← o-entity(x,w_1),
o-entity(x,w_2),
...
o-entity(x,w_t),
relationship-role-description(x,r_1,p_1),
relationship-role-description(x,r_2,p_2),
...
relationship-role-description(x,r_u,p_u),
relationship-attribute-description(x,a_1,v_1),
relationship-attribute-description(x,a_2,v_2),
...
relationship-attribute-description(x,a_v,v_v).

Rule for the description of entity sets:

26. Given an entity set 'y' with mandatory attributes 'a_1', 'a_2', ..., 'a_p' associated with value sets '$vset_1$', '$vset_2$', ..., '$vset_p$', mandatory parts from the entity sets '$eset_1$', '$eset_2$', ..., '$eset_q$', and mandatory organizing relationships from the relationship sets '$rset_1$', '$rset_2$', ..., '$rset_r$', create the following clause that must be

satisfied by any entity 'X' to become its member:

entity-set-member(X,y) ←
entity-attribute-description(X,a_1,V_1),
value-set-member(V_1,$vset_1$),
entity-attribute-description(X,a_2,V_2),
value-set-member(V_2,$vset_2$),
... ...
entity-attribute-description(X,a_p,V_p),
value-set-member(V_p,$vset_p$),
p-entity(X,E_1),
entity-set-member(E_1,$eset_1$),
p-entity(X,E_2),
entity-set-member(E_2,$eset_2$),
... ...
p-entity(X,E_q),
entity-set-member(E_q,$eset_q$),
o-relationship(X,R_1),
relationship-set-member(R_1,$rset_1$),
o-relationship(X,R_2),
relationship-set-member(R_2,$rset_2$),
... ...
o-relationship(X,R_r),
relationship-set-member(R_r,$rset_r$).

Rule for the description of relationship sets:

27. Given a relationship set 'y' containing relationships that form entities in the entity sets '$eset_1$', '$eset_2$', ..., '$eset_u$' mandatorily, with its associated roles 'r_1', 'r_2', ..., 'r_v' being played by the entities from the entity sets 'est_1', 'est_2', ..., 'est_v' mandatorily, and having mandatory attributes 'a_1', 'a_2', ..., 'a_w' drawing values from the value sets '$vset_1$', '$vset_2$', ..., '$vset_w$', create the following clause for any relationship 'X' to become its member:

relationship-set-member(X,y) ←
o-entity(X,Z_1),
entity-set-member(Z_1,$eset_1$),

o-entity(X,Z_2),
entity-set-member(Z_1,$eset_2$),
... ...
o-entity(X,Z_u),
entity-set-member(Z_u,$eset_u$),
relationship-role-description(X,a_1,E_1),
entity-set-member(E_1,$eset_1$),
relationship-role-description(X,a_2,E_2),
entity-set-member(E_2,$eset_2$),
... ...
relationship-role-description(X,a_v,E_v),
entity-set-member(E_v,$eset_v$),
relationship-attribute-description(X,a_1,V_1),
value-set-member(V_1,$vset_1$),
relationship-attribute-description(X,a_2,V_2),
value-set-member(V_2,$vset_2$),
... ...
relationship-attribute-description(X,a_w,V_w),
value-set-member(V_w,$vset_w$).

Using the above rules, a given set of generic, instance, and membership ER-diagrams are converted into a set of facts and rules. As an example, the following is obtained from the ER-diagrams in figures 2.12, 2.13, 2.14, 2.15, and 2.16 by applying the above translation rules in the order in which they are listed:

entity(rb1).
entity(rp1).
entity(rp2).
entity(rp3).
entity(rp4).
entity(table24).
entity(book51).
entity(study-room2).

relationship(support1).
relationship(support1).
relationship(support2).
relationship(support3).
relationship(support4).
relationship(keep5).

role(supporter).

role(supportee).
role(instrument).
role(object).

entity-attribute(top).
entity-attribute(legs).
entity-attribute(weight).
entity-attribute(length).
entity-attribute(width).
entity-attribute(height).
entity-attribute(color).

relationship-attribute(strength).

value(1).
value(4).
value(14).
value(48).
value(20).
value(brown).
value(good).

entity-set(rectangular-block).
entity-set(rectangular-prism).
entity-set(table).
entity-set(book).
entity-set(study-room).

relationship-set(support).
relationship-set(keep).

value-set(num-tops).
value-set(num-legs).
value-set(table-weights).
value-set(table-lengths).
value-set(table-widths).
value-set(table-heights).
value-set(color-names).
value-set(quality).

entity-set-member(rb1,rectangular-block).
entity-set-member(rp1,rectangular-prism).
entity-set-member(rp2,rectangular-prism).
entity-set-member(rp3,rectangular-prism).
entity-set-member(rp4,rectangular-prism).
entity-set-member(table24,table)
entity-set-member(book51,book).
entity-set-member(study-room2,study-room).

relationship-set-member(support1,support).

relationship-set-member(support2,support).
relationship-set-member(support3,support).
relationship-set-member(support4,support).
relationship-set-member(keep5,keep).

value-set-member(1,num-tops).

value-set-member(4,num-legs).

value-set-member(black,color-names).

value-set-member(blue,color-names).

value-set-member(brown,color-names).

value-set-member(green,color-names).

value-set-member(orange,color-names).

value-set-member(red,color-names).

value-set-member(yellow,color-names).

value-set-member(X,table-weights) $\leftarrow$ $X \geq 1, X \leq 50.$

value-set-member(X,table-lengths) $\leftarrow$ $X \geq 2, X \leq 10.$

value-set-member(X,table-widths) $\leftarrow$ $X \geq 2, X \leq 8.$

value-set-member(X,table-heights) $\leftarrow$ $X \geq 1, X \leq 4.$

entity-attribute-description(table24,top,1).
entity-attribute-description(table24,legs,4).
entity-attribute-description(table24,weight,14).
entity-attribute-description(table24,length,48).
entity-attribute-description(table24,width,48).
entity-attribute-description(table24,height,20).
entity-attribute-description(table24,color,brown).

entity-role-description(rb1,supportee,support1).
entity-role-description(rb1,supportee,support2).
entity-role-description(rb1,supportee,support3).
entity-role-description(rb1,supportee,support4).
entity-role-description(rp1,supporter,support1).
entity-role-description(rp2,supporter,support2).
entity-role-description(rp3,supporter,support3).
entity-role-description(rp4,supporter,support4).
entity-role-description(table24,instrument,keep5).
entity-role-description(book51,object,keep5).

w-entity(rb1,table24).
w-entity(rp1,table24).
w-entity(rp2,table24).
w-entity(rp3,table24).

w-entity(rp4,table24).
w-entity(table24,study-room2).
w-entity(book51,study-room2).

p-entity(X,Y) ← w-entity(Y,X).

o-relationship(table24,support1).
o-relationship(table24,support2).
o-relationship(table24,support3).
o-relationship(table24,support4).
o-relationship(study-room2,keep5).

entity-description(X) ← entity-external-description(X),
entity-internal-description(X).

entity-external-description(rb1) ← entity-role-description(rb1,supportee,support1),
entity-role-description(rb1,supportee,support2),
entity-role-description(rb1,supportee,support3),
entity-role-description(rb1,supportee,support4),
w-entity(rb1,table24).

entity-external-description(rp1) ← entity-role-description(rp1,supportee,support1),
w-entity(rp1,table24).

entity-external-description(rp2) ← entity-role-description(rp2,supportee,support2),
w-entity(rp2,table24).

entity-external-description(rp3) ← entity-role-description(rp3,supportee,support3),
w-entity(rp3,table24).

entity-external-description(rp4) ← entity-role-description(rp4,supportee,support4),
w-entity(rp4,table24).

entity-external-description(table24) ← entity-attribute-description(table24,top,1),
entity-attribute-description(table24,legs,4),
entity-attribute-description(table24,weight,14),
entity-attribute-description(table24,length,48),
entity-attribute-description(table24,width,4),
entity-attribute-description(table24,height,4),
entity-attribute-description(table24,color,brown),
entity-role-description(table24,instrument,keep5),
w-entity(table24,study-room2).

entity-external-description(book51) ← entity-role-description(book51,object,keep5),
w-entity(book51,study-room2).

entity-external-description(study-room2) ← .

entity-internal-description(rb1) ←.

entity-internal-description(rp1) ←.

entity-internal-description(rp2) ←.

```
entity-internal-description(rp3) ←.

entity-internal-description(rp4) ←.

entity-internal-description(table24) ← p-entity(table24,rb1),
                                       p-entity(table24,rp1),
                                       p-entity(table24,rp2),
                                       p-entity(table24,rp3),
                                       p-entity(table24,rp4),
                                       o-relationship(table24,support1),
                                       o-relationship(table24,support2),
                                       o-relationship(table24,support3),
                                       o-relationship(table24,support4).

entity-internal-description(book51) ←.

entity-internal-description(study-room2) ←
                                       p-entity(study-room2,table24),
                                       p-entity(study-room2,book51),
                                       o-relationship(study-room2,keep5).

o-entity(X,Y)      ←    o-relationship(Y,X).

relationship-role-description(X,Y,Z) ←    entity-role-description(Z,Y,X).

relationship-attribute-description(support1,strength,good).
relationship-attribute-description(support2,strength,good).
relationship-attribute-description(support3,strength,good).
relationship-attribute-description(support4,strength,good).

relationship-description(support1) ← o-entity(support1,table24),
                                     relationship-role-description(support1,supporter,rp1),
                                     relationship-role-description(support1,supportee,rb1),
                                     relationship-attribute-description(support1,strength,good).

relationship-description(support2) ← o-entity(support2,table24),
                                     relationship-role-description(support2,supporter,rp2),
                                     relationship-role-description(support2,supportee,rb1),
                                     relationship-attribute-description(support2,strength,good).

relationship-description(support3) ← o-entity(support3,table24),
                                     relationship-role-description(support3,supporter,rp3),
                                     relationship-role-description(support3,supportee,rb1),
                                     relationship-attribute-description(support3,strength,good).

relationship-description(support4) ← o-entity(support4,table24),
                                     relationship-role-description(support4,supporter,rp4),
                                     relationship-role-description(support2,supportee,rb1),
                                     relationship-attribute-description(support2,strength,good).

relationship-description(keep5) ← o-entity(keep5,study-room2),
                                  relationship-role-description(keep5,instrument,table24),
```

relationship-role-description(keep5,object,book51).

entity-set-member(X,table) ←
entity-attribute-description(X,top,V_1),
value-set-member(V_1,num-tops),
entity-attribute-description(X,legs,V_2),
value-set-member(V_2,num-legs),
entity-attribute-description(X,weight,V_3),
value-set-member(V_3,table-weights),
entity-attribute-description(X,length,V_4),
value-set-member(V_4,table-lengths),
entity-attribute-description(X,width,V_5),
value-set-member(V_5,table-widths),
entity-attribute-description(X,height,V_6),
value-set-member(V_6,table-heights),
entity-attribute-description(X,color,V_7),
value-set-member(V_7,color-names),
p-entity(X,E_1),
entity-set-member(E_1,rectangular-block),
p-entity(X,E_2),
entity-set-member(E_2,rectangular-prism),
p-entity(X,E_3),
entity-set-member(E_3,rectangular-prism),
p-entity(X,E_4),
entity-set-member(E_4,rectangular-prism),
p-entity(X,E_5),
entity-set-member(E_5,rectangular-prism),
o-relationship(X,R_1),
relationship-set-member(R_1,support),
o-relationship(X,R_2),
relationship-set-member(R_2,support),
o-relationship(X,R_3),
relationship-set-member(R_3,support),
o-relationship(X,R_4),
relationship-set-member(R_4,support).

entity-set-member(X,study-room) ←
p-entity(X,E_1),
entity-set-member(E_1,table),
p-entity(X,E_2),
entity-set-member(E_2,book),
o-relationship(X,R_1),
relationship-set-member(R_1,keep).

relationship-set-member(X,support) ←
o-entity(X,Z_1),
entity-set-member(Z_1,table),
relationship-role-description(X,supportee,E_1),
entity-set-member(E_1,rectangular-block),
relationship-role-description(X,supporter,E_2),
entity-set-member(E_2,rectangular-prism),
relationship-attribute-description(X,strength,V_n),
value-set-member(V_n,quality).

relationship-set-member(X,keep) ←
o-entity(X,Z_1),
entity-set-member(Z_1,study-room),
relationship-role-description(X,instrument,E_1),
entity-set-member(E_1,table),
relationship-role-description(X,object,E_2),
entity-set-member(E_2,book).

The above symbolic data structures (facts and rules) obtained from the translation of ER-diagrams are loaded into a knowledge base management system capable of maintaining the production system or the first-order logic system being implemented. If an OPS5 interpreter [For81] is used to drive the above set of facts and rules, then we obtain a production system. If the facts and the rules are driven by a PROLOG interpreter [Per84], then the resulting system becomes a first-order logic system.

CHAPTER IV

A Frame-Based Implementation Of An ER-Knowledge Base System

4.1. Introduction

In this chapter, we are concerned with the implementation of a frame-based knowledge base system from the specifications provided by Entity-Relationship approach. First we will examine frame-based systems and then show how to guide the implementation of a frame-based system from the specifications obtained using Entity-Relationship approach. Translation rules for the conversion of ER-diagrams into a frame-based representation are provided by mapping appropriate symbolic data structures.

4.2. Schema

Frame-based systems are based on the notion of a schema [Bar32]. Bartlett introduced the idea of a schema to explain how people remember situations that they encountered previously. Minsky proposed *frame* as a data structure to represent a schema [Min75]. Schema-based implementations are called by a variety of names such as frame-based systems [FiK85, KLC87, Kui75, RoG77], and script-based systems [ScA77]. In this dissertation, we will use the terms *schema* and *frame* interchangeably.

4.3. A Frame Notation

A frame is a data structure for representing a stereotyped situation [Min75]. A frame-based system simulates expectant or predictive behavior [BaF81]. Important frame-based systems include FRL [RoG77], KRL [BoW77], KL-ONE [Bra78], and NETL [Fah79].

4.3.1. Slots of a Frame

Every frame has a name and a number of slots that capture the description of the situation that it represents. The general format of a frame is the following [KaC87, WiH84]:

```
(<frame name>   (<slot1>  (<facet1>  (<value1>  <value2> ... ))
                          (<facet2>  (<value1>  <value2> ... ))
                                                    ...          )

                (<slot2>  (<facet1>  (<value1>  <value2> ... ))
                          (<facet2>  (<value1>  <value2> ... ))
                                                    ...          )
                    ...                             ...              )
```

The following is an example frame that represents the details of a person named 'henry' [WiH84]:

```
(henry      (a-kind-of      (value          (man)))
            (height         (value          (1.78)))
```

(weight (value (75)))

(hobbies (value (jogging skiing))))

The above example indicates that each slot in a frame represents a particular property of the thing being described.

4.3.2. Facets of a Slot

A slot may have more than one facet. While a slot represents a specific property of a thing, facets of a slot allow the representation of different types of values that the property being described may have. Thus, facets introduce the possibility of describing the value(s) of a given property in different ways. Three types of facets popular in frame-based systems are value facets, default facets, and demons.

4.3.2.1. Value Facets

When the actual value of a property being described is known, it is represented by a facet with the name 'value'. Consider the above example frame 'henry' in which his actual weight is indicated by the following 'value' facet:

(henry

(weight (value (75)))

4.3.2.2. Default Facets

When actual values of a particular instance is not known, default values of the generic class to which the instance belongs may be assumed. For example, when henry's weight is not known, it is reasonable to assume his weight to be that of an

average man, say 70 kgs. Default values of a slot are represented in the facet named 'default'. The following represents henry's default weight:

```
(henry
        (weight        (default    (70)))
```

4.3.2.3. Demons

In addition to the incorporation of actual and default values, facets also permit the possibility of attaching procedures to a slot in a frame. Such procedures are known as the 'demons'. A frame-based system usually provides the three demons: 'if-needed', 'if-added', and 'if-deleted'. An if-needed demon is a procedure that computes the value of the property represented by a slot when needed. The following is an example of a if-needed demon:

```
(henry
    (weight     (if-needed (* fget(henry,volume) fget(henry,density)))))
```

The if-needed demon attached to the slot 'weight' in the frame 'henry' when invoked computes the weight of henry by accessing and multiplying the values of his volume and density. Note that 'fget' is an access procedure provided by the frame-based system. The two demons 'if-added' and 'if-removed' monitor the slot to which they are attached and act when a value is added or removed from the slot.

4.3.3. Need to Distinguish Different Types of Slots and Frames

Despite its richness of representation, a slot treats every description uniformly as a property of the frame. However, we have seen in chapter II that different types of descriptions of a thing such as its attributes, parts, wholes, roles and organizing relationships must be distinguished. Therefore, we will introduce different types of slots to distinguish different kinds of descriptions in sections 4.3.4 through 4.3.5. Having different types of slots increases the expressiveness of a frame similar to different types of facets increasing the expressiveness of a slot in a frame.

In a frame-based system, all frames are treated uniformly. They do not distinguish whether a given frame is describing an object or a complex relationship. Therefore, we introduce different types of frames to explicitly represent entities, relationships, entity sets, and relationship sets. We introduce two types of instance frames, E-frames and R-frames, to represent individual entities and relationships respectively. The two types of generic frames, ES-frames and RS-frames, are introduced to represent entity sets and relationship sets respectively.

The following sections will describe the different types of frames needed to represent entities, relationships, entity sets, and relationship sets. First we will consider the instance frames and then we describe the generic frames.

4.3.4. Instance Frames

A frame that represents a specific instance is called an instance frame. Using the notation in chapter II, we can focus our attention on the parts themselves, or the organization of the parts, or the wholes that are formed. Instances of both parts and

wholes are represented by entities. Relationships represent instances of organizations of parts in wholes. In this section, we introduce two types of instance frames that represent individual entities and relationships respectively.

4.3.4.1. E-frames

The instance frames used to represent individual entities are called E-frames. For each entity a corresponding E-frame is created. The name and description of the E-frame created correspond to the name and the description of the entity it represents. For example, the E-frame that represents the entity 'table24' in figure 2.12 is named 'table24'. Any frame preceded by the symbol '*' is treated as an E-frame as shown in the examples provided in this section.

Since an entity can have different types of descriptions, we introduce different types of slots to represent them explicitly. The slots that represent entity attributes are preceded by the symbol '@'. A slot representing an entity attribute has the same name as that of the entity attribute. For example, the attribute 'length' of the entity 'table24' having a value of 48" in figure 2.12 is represented as follows:

*(table24

(@length (value (48"))))

The roles that an entity plays are represented by the slots preceded by the symbol '#'. The entity 'table24' playing the role 'instrument' in the relationship 'keep5' in figure 2.12 is represented as follows:

*(table24

(#instrument (value (keep5))))

The wholes in which an entity participates are represented by a slot with the name 'w-entities'. The fact that 'table24' being a part of the whole 'study-room2' is represented as follows:

*(table24

(w-entities (value (study-room2))))

The parts of an entity are represented by a slot with the name 'p-entities'. The 'table24', for example, having the rectangular block 'rb1', and the rectangular prisms 'rp1', 'rp2', 'rp3', and 'rp4' as its parts is represented as below:

*(table24

(p-entities (value ((rb1), (rp1, rp2,rp3, rp4)))))

The relationships that provide organization to an entity are represented by a slot with the name 'o-relationships'. The relationships 'support1', 'support2', 'support3', and 'support4' providing the organization to the 'table24' in figure 2.12 are represented as follows:

*(table24

(o-relationships (value (support1, support2, support3, support4))))

In addition to providing the description of an entity, an E-frame also indicates the entity set(s) to which the entity described belongs to. The slot 'member-of' in an E-frame indicates the ES-frames (section 4.3.5.1) that represent the entity sets in which the entity represented by the E-frame belongs to. For example, the entity 'table24' being a member of the entity set 'table' (figure 2.14) is represented as follows:

*(table24

(member-of (value (table))))

Note that the description that an E-frame provides does not include default values. Defaults are represented in the ES-frame representing the entity set in which the entity represented by the E-frame is a member. They are discussed in section 4.3.5.1.

4.3.4.2. R-frames

R-frames are the instance frames that are used to represent individual relationships. For each relationship a corresponding R-frame is created. The name and the description of the R-frame created correspond to the name and the description of the relationship that it represents. The R-frame that represents the relationship 'support1' in figure 2.12, for example, is named 'support1'. Every R-frame is preceded by the symbol '&' as shown in the examples provided in this section.

In order to represent the different types of descriptions of a relationship explicitly, we introduce different types of slots. The slot with the name 'o-entities' in a R-frame represents the whole(s) that a relationship represented by the R-frame helps form. The relationship 'support1' in figure 2.12 providing organization to the whole 'table24' is represented as follows:

&(support1

(o-entities (value (table24))))

A relationship should explicitly state the roles performed by each of the entities participating in it. The slots that represent the roles of a relationship are preceded by

the symbol '#'. The role 'supportee' in the relationship 'support1' in figure 2.12 played by entity 'rb1' is represented as follows:

&(support1

(#supportee (value (rb1))))

The slots that represent the attributes of a relationship are preceded by the symbol '@'. For example, the 'support1' relationship having the attribute 'strength' with a value 'good' is represented below:

&(support1

(@strength (value (good))))

In addition to providing the description of a relationship, a R-frame also indicates the relationship set(s) to which the relationship described belongs to. The slot 'member-of' in an R-frame indicates the RS-frames (section 4.3.5.2) that represent the relationship sets in which the relationship represented by the R-frame belongs to. For example, the relationship 'support1' being a member of the relationship set 'support' (figure 2.15) is represented as follows:

&(support1

(member-of (value (support))))

Our description of R-frames included only the known values of a relationship. Default values of a relationship are included in the RS-frame representing the relationship set to which the relationship represented by the R-frame belongs. The default values of relationships are described in section 4.3.5.2.

4.3.5. Generic Frames

Frames that represent generic classes are called generic frames. A generic frame serves several purposes. First, its description provides the criteria that an instance must satisfy to become a member of the generic class it represents. It also includes the description of a prototype to provide default descriptions to the instances of the class it represents. Furthermore, it indicates the general class(es) to which the given class belongs to. In this section, we introduce two types of generic frames that represent entity sets and relationship sets respectively.

4.3.5.1. ES-frames

An ES-frame is a generic frame that represents an entity set. For each entity set a corresponding ES-frame is created. The name and the description of a ES-frame correspond to the name and the description of the entity set it represents. For example, the ES-frame that represents the entity set 'table' in figure 2.13 is named 'table'. Every ES-frame is preceded by the symbol '**' as shown in the examples provided in this section.

As mentioned in section 2.6.3.1, the description of an entity set specifies the criteria that must be satisfied by an entity to become one of its members. Therefore, an ES-frame should provide the criteria that E-frames of entities in the entity set represented by it must satisfy. Since an entity set has different types of descriptions, we introduce different types of slots to represent them explicitly. Furthermore, different types of descriptions of an entity set may be mandatory, optional, or forbidden. Therefore, we introduce three corresponding facets called 'mandatory', 'optional', and

'forbidden' facets.

For each of its attributes, an entity set specifies a value set from which each of the entities in it assume values. Attributes are represented in a ES-frame by the slots preceded by the symbol '@'. The attribute 'length' of the entity set 'table' in figure 2.13 associated with the value set 'table-lengths' is represented as follows:

**(table

(@length (mandatory (table-lengths))))

An entity set is associated with a relationship set for each of the roles that the entities in it play. The slots that indicate the role information in an entity set are preceded by the symbol '#'. The following represents the relationship set 'keep' being associated with the role 'instrument' of the entity set 'table':

**(table

(#instrument (optional (keep))))

The entity sets representing the wholes in which the entities in a given entity set participate are represented by the slot named 'w-entities'. The entity set 'table' having the entity set 'study-room' as one of its 'w-entities' is represented as below:

**(table

(w-entities (optional (study-room))))

The entity sets containing the parts of the entities in an entity set is represented by the slot named 'p-entities'. The following represents the entity set 'table' having the entity sets 'rectangular-block (rb)' and 'rectangular-prism (rp)' as its p-entities:

**(table

(p-entities (mandatory (rb, rp))))

The relationship sets that contain relationships providing organization to the parts of the entities in an entity set are represented by the slot 'o-relationships'. The following represents that the entities in the entity set 'table' are formed by the organization provided by the relationships in the relationship set 'support':

**(table

(o-relationships (mandatory (support))))

An ES-frame can also be used to represent the prototype of the entity set that it represents. The prototype of an entity set provides the default descriptions to the entities in the entity set when specific details are not known. For example, if the actual length of a particular table is not known, the length of the prototype table mentioned in the following default facet of the entity set 'table' is assumed:

**(table

(@length (default (48"))))

The following description of the ES-frame representing the entity set 'table' includes default information provided by the table prototype:

**(table

(@top (mandatory (num-tops))

(default (1)))

(@legs (mandatory (num-legs))

(default (4)))

(@weight (mandatory (table-weights))

```
                    (default     (14)))
(@length            (mandatory (table-lengths))
                    (default     (48")))
(@width             (mandatory (table-widths))
                    (default     (48")))
(@height            (mandatory (table-heights))
                    (default     (20")))
(@color             (mandatory (color-names))
                    (default     (brown)))
(#instrument        (optional    (keep))
                    (default     (keep5)))
(w-entities         (optional    (study-room))
                    (default     (study-room2)))

(p-entities         (mandatory (rb, rp))
                    (default     ((rb1,rp1,rp2,rp3,rp4), (rp1))))

(o-relationships    (mandatory (support))
                    (default    ((support1, support2, support3, support4))))
)
```

4.3.5.2. RS-frames

The generic frames that represent relationship sets are called RS-frames. For each relationship set a corresponding RS-frame is created. The name and the descrip-

tion of a RS-frame correspond to the name and the description of the relationship set it represents. For example, the RS-frame that represents the relationship set 'support' in figure 2.13 is named 'support'. Every RS-frame is preceded by the symbol '&&' as shown in the examples provided in this section.

A relationship set specifies the criteria that must be satisfied by a relationship to become one of its members. Thus, a RS-frame should provide the criteria that R-frames of relationships in the relationship set must satisfy. Since a relationship set has different types of descriptions, we introduce different types of slots to represent them explicitly. Furthermore, different types of descriptions of a relationship set may be mandatory, optional, or forbidden. Therefore, we use the three corresponding facets called 'mandatory', 'optional', and 'forbidden' facets.

The entity sets representing the wholes formed by the organization provided by the relationships in a relationship set are indicated by the slot name 'o-entities'. The relationship set 'support' in figure 2.13 having the entity set 'table' as one of its 'o-entities' is represented below:

```
&&(support
                (o-entities        (mandatory (table))))
```

Each of the roles of a relationship set is associated an entity set. The slots that indicate the role information in a relationship set are preceded by the symbol '#'. The following represents the entity set 'rectangular block (rb)' being associated with the role 'supportee' of the relationship set 'support':

```
&&(support
                (@supportee        (mandatory (rb))))
```

For each of its attributes, a relationship set specifies a value set from which each of the relationships in it assume values. Attributes are represented in a RS-frame by the slots preceded by the symbol '@'. The attribute 'strength' of the relationship set 'support' in figure 2.13 associated with the value set 'quality' is represented as follows:

```
&&(support
        (@length        (mandatory (quality))))
```

A RS-frame can also be used to represent a prototype of the relationship set that it represents. The prototype of a relationship set provides the default descriptions to the relationships in the relationship set when specific details are not known. For example, if the actual strength of particular relationship is not known the strength of the prototype relationship 'support' mentioned in the following default facet of the relationship 'support' is assumed:

```
&&(support
        (@length        (default    (good))))
```

The following description of the RS-frame representing the relationship set 'support' includes default information provided by the support prototype:

```
&&(support
    (o-entities         (mandatory (table))
                        (default    (table24)))
    (#supportee (mandatory (rb))
                        (default    (rb1)))
```

```
(#supporter     (mandatory (rp))
                (default   (rp1)))
(@strength      (mandatory (good))
                (default   (study-room2)))
)
```

4.4. Translation Rules for the Conversion of ER-diagrams:

The following provides the summary of the translation rules proposed in this chapter for the conversion of ER-diagrams into a frame-based representation:

Rules for the representation of entities:

1. Create an **E-frame** for each entity.
2. Create a **@attribute-name** slot in the E-frame of an entity for each of its attributes. List the attribute value(s) in the value fact of the slot created.
3. Create a **#role-name** slot in the E-frame of an entity for each role it plays. List the relationship(s) in which the given entity plays the role under consideration in the value facet of the slot created.
4. Create the **w-entities** slot in the E-frame of each entity. List the entity (or entities) in which the given entity is a part in the value facet of the slot created.
5. Create the **p-entities** slot in the E-frame of each entity. List the entity (or entities) that are part of the given entity in the value facet of the slot created.
6. Create the **o-relationships** slot in the E-frame of each entity. List the relationship(s) that organize the parts of the given entity in the value facet of the

slot created.

7. Create the **member-of** slot in the E-frame of each entity. List the entity set(s) in which the given entity is a member in the value facet of the slot created.

Rules for the representation of relationships:

8. Create a **R-frame** for each relationship.
9. Create the **o-entities** slot in the R-frame of each relationship. List the entity (or entities), in which the given relationship provides the organization to their component entities, in the value facet of the slot created.
10. Create a **#role-name** slot in the R-frame of a relationship for each of the roles associated with it. List the entity (or entities) that play the role of the relationship under consideration in the value facet of the slot created.
11. Create a **@attribute-name** slot in the R-frame of a relationship for each of its attributes. List the attribute value(s) in the value facet of the slot created.
12. Create the **member-of** slot in the R-frame of each relationship. List the relationship set(s) in which the given relationship is a member in the value facet of the slot created.

Rules for the representation of entity sets:

13. Create an **ES-frame** for each entity set.
14. Create a **@attribute-name** slot in the ES-frame of an entity set for each of the attributes associated with it. List the value set associated with the attribute under consideration in the mandatory facet of the slot created. List the attribute value(s)

of the prototype of the entity set in the default facet of the slot created.

15. Create a **#role-name** slot in the ES-frame of an entity set for each of the roles that the entities in it may play. List the relationship set(s) associated with the role under consideration in the optional facet of the slot created. List the relationship(s), in which the prototype of the entity set plays the given role, in the default facet of the slot created.

16 Create the **w-entities** slot in the ES-frame of each entity set. List the entity set(s), in which the entities in the given entity set may become parts of the entities in the entity sets listed, in the optional facet of the slot created. List the whole(s) in which the prototype of the entity set participates in the default facet of the slot created.

17. Create the **p-entities** slot in the ES-frame of each entity set. List the entity set(s) that contain the parts of the entities in the given entity set in the mandatory facet of the slot created. List the parts that make up the prototype of the entity set in the default facet of the slot created.

18. Create the **o-relationships** slot in the ES-frame of each entity set. List the relationship set(s) containing the relationships that organize the parts of the entities in the given entity set in the mandatory facet of the slot created. List the relationship(s) that organize the parts of the prototype of the entity set in the default facet of the slot created.

Rules for the representation of relationship sets:

19. Create a **RS-frame** for each relationship set.

20. Create the **o-entities** slot in the RS-frame of each relationship set. List the entity set(s) that contain the entities whose parts are organized by the relationships in the given relationship set in the mandatory facet of the slot created . List the entity (or entities) whose parts are organized by the prototype of the relationship set in the default facet of the slot created.

21. Create a **#role-name** slot in the RS-frame of a relationship set for each of the roles associated with it. List the entity set(s) containing the entities that play the role under consideration in the relationships of the relationship set in the mandatory facet of the slot created. List the entity (or entities) that play the given role in the prototype of the relationship set in the default facet of the slot created.

22. Create a **@attribute-name** slot in the RS-frame of a relationship set for each of its attributes. List the value set associated with the attribute under consideration in the mandatory facet of the slot created. List the attribute value(s) of the prototype of the relationship set in the default facet of the slot created.

The above rules are applied to a given set of generic, instance, and membership ER-diagrams to obtain a set of frames. The following is obtained when the translation rules are applied to the figures 2.12, 2.13, 2.14, 2.15, and 2.16.

```
*(rb1
      (#supportee      (value      (support1,support2,support3,support4)))
      (w-entities      (value      (table24)))
      (member-of       (value      (rb)))
)
```

```
*(rp1
      (#supporter       (value      (support1)))
      (w-entities       (value      (table24)))
      (member-of        (value      (rp)))
)

*(rp2
      (#supporter       (value      (support2)))
      (w-entities       (value      (table24)))
      (member-of        (value      (rp)))
)

*(rp3
      (#supporter       (value      (support3)))
      (w-entities       (value      (table24)))
      (member-of        (value      (rp)))
)

*(rp4
      (#supporter       (value      (support4)))
      (w-entities       (value      (table24)))
      (member-of        (value      (rp)))
)

*(table24
      (@top             (value      (1)))
      (@legs            (value      (4)))
      (@weight          (value      (14lb)))
      (@length          (value      (48")))
      (@breadth         (value      (48")))
      (@height          (value      (20")))
      (@color           (value      (brown)))
      (#instrument      (value      (keep5)))
      (w-entities       (value      (study-room2)))
      (p-entities       (value      ((rb1), (rp1, rp2, rp3, rp4))))
      (o-relationships  (value      (support1, support2, support3, support4)))
      (member-of        (value      (table)))
)

*(book51
      (#object          (value      (keep5)))
      (w-entities       (value      (study-room2)))
      (member-of        (value      (book)))
)
```

```
*(study-room2
      (p-entities          (value      ((table24) (book51))))
      (o-relationship      (value      (keep5)))
      (member-of           (value      (study-room)))
)

&(support1
      (o-entities          (value      (table24)))
      (#supportee          (value      (rb1)))
      (#supporter          (value      (rp1)))
      (@strength           (value      (good)))
      (member-of           (value      (support)))
)

&(support2
      (o-entities          (value      (table24)))
      (#supportee          (value      (rb1)))
      (#supporter          (value      (rp2)))
      (@strength           (value      (good)))
      (member-of           (value      (support)))
)

&(support3
      (o-entities          (value      (table24)))
      (#supportee          (value      (rb1)))
      (#supporter          (value      (rp3)))
      (@strength           (value      (good)))
      (member-of           (value      (support)))
)

&(support4
      (o-entities          (value      (table24)))
      (#supportee          (value      (rb1)))
      (#supporter          (value      (rp4)))
      (@strength           (value      (good)))
      (member-of           (value      (support)))
)

$(keep5
      (o-entities          (value      (study-room5)))
      (#instrument         (value      (table24)))
      (#object             (value      (book51)))
      (member-of           (value      (keep)))
)
```

```
**(rb
      (#supportee       (optional    (support))
                        (default     (support1)))
      (w-entities       (optional    (table))
                        (default     (table24)))
)

**(rp
      (#supporter       (optional    (support))
                        (default     (support1)))
      (w-entities       (optional    (table))
                        (default     (table24)))
)

**(table
      (@top             (mandatory (num-tops))
                        (default     (1)))
      (@legs            (mandatory (num-legs))
                        (default     (4)))
      (@weight          (mandatory (table-weights))
                        (default     (14)))
      (@length          (mandatory (table-lengths))
                        (default     (48")))
      (@width           (mandatory (table-widths))
                        (default     (48")))
      (@height          (mandatory (table-heights))
                        (default     (20")))
      (@color           (mandatory (color-names))
                        (default     (brown)))
      (#instrument      (optional    (keep))
                        (default     (keep5)))
      (w-entities       (optional    (study-room))
                        (default     (study-room2)))
      (p-entities       (mandatory (rb, rp))
                        (default     ((rb1,rp1,rp2,rp3,rp4), (rp1))))
      (o-relationships  (mandatory (support))
                        (default ((support1, support2, support3, support4))))
)

**(book
      (#object          (optional    (keep))
                        (default     (keep5)))
      (w-entities       (optional    (study-room))
                        (default     (study-room2)))
)
```

```
**(study-room
      (p-entities          (mandatory (table book))
                           (default   ((table24) (book51))))
      (o-relationships     (mandatory (keep))
                           (default   (keep5)))
)

&&(support
      (o-entities          (mandatory (table))
                           (default   (table24)))
      (#supportee          (mandatory (rb))
                           (default   (rb1)))
      (#supporter          (mandatory (rp))
                           (default   (rp1)))
      (@strength           (mandatory (good))
                           (default   (study-room2)))
)

&&(keep
      (o-entities          (mandatory (study-room))
                           (default   (study-room2)))
      (#instrument         (mandatory (table))
                           (default   (table24)))
      (#object             (mandatory (book))
                           (default   (book51)))
)
```

The above frame data structures obtained from the translation of the ER-diagrams are loaded into a knowledge base management system capable of maintaining the frame-based system being implemented. For example, they may be loaded into a frame management system such as FRL [RoG77]. Since most frame management systems do not distinguish the different types of frames and slots dealt here, they need to be modified to incorporate the maintenance of these features.

CHAPTER V

Representation of Surface and Deep Structures in Entity-Relationship Approach

5.1. Introduction

Knowledge base systems often provide a front-end that supports user interaction in a natural language. In this chapter, we will be concerned with using Entity-Relationship approach for natural language analysis. Since sentences in natural language have surface and deep structures, we will examine how they can be represented in Entity-Relationship approach. The correspondence between sentence structures and Entity-Relationship approach also helps the translation of documents written in a natural language into ER-diagrams [Che83].

5.2. Sentence Structure

There is a general agreement that sentences in natural language have two kinds of structures: 'surface structure' and 'deep structure' [Cho65, Fil68, FoH78, HaC83, Win83]. The surface structure of a sentence is governed by the rules of grammar. Consider the sentence 'John is willing to help'. 'John' is a noun, 'is' a verb, 'willing' an adjective, 'to' a preposition, and 'help' a verb. This simple analysis gives the sentence's surface structure, put together by the rules of English grammar.

It has been observed that sentences also have a deep structure, which in many cases is not the same as the surface structure [Cho65, Fil68]. For example, consider

the two sentences, the one mentioned above and another: 'John is difficult to help'. In terms of surface grammar, the two sentences are alike. The only difference between them is the word 'willing' in one sentence and the world 'difficult' in another. Both are adjectives, and both are in the same position in the sentence. A further analysis, however, reveals that there are important semantic differences between the two sentences. In 'John is willing to help', John is the person doing the help. In 'John is difficult to help', John is the person to be helped. Technically, in the first sentence, John is the subject of the verb 'help'; in the second sentence he is the object.

5.3. Surface Structure

Structure of a sentence that conforms to a syntactic grammar is known as the surface structure of the sentence. Pure syntactic approaches to natural language processing rely on a syntactic grammar that determines whether a sentence is legal or not. A sentence is treated as a string of words, each classified according to its function into lexical categories called the parts of speech. In this section we will first examine the parts of speech in English language and their correspondence to the primitives provided by the Entity-Relationship approach. We will then discuss the role of grammar in parsing sentences and converting them into ER-diagrams.

5.3.1. Parts of Speech

Parts of speech is a term used to describe the class of words to which a particular word belongs according to its function in a sentence. Each function in a sentence is performed by a word belonging to a certain part of speech. Major parts of speech

include nouns, pronouns, verbs, adjectives, adverbs, and prepositions. The following subsections explain what they are and also discuss what they correspond to in Entity-Relationship Approach. The idea of relating parts of speech in English with ER-diagrams was originally proposed by Chen [Che83]. We will extend it to different types of verbs, articles, and prepositions. Our objective here is to provide a detailed discussion of given a word how to classify it into a particular type of parts of speech and then how to represent it in an ER-diagram.

5.3.1.1. Nouns and Pronouns

If the function of a word is to name something, then it is called a noun or pronoun. Usually nouns are used to name people, places, things, or concepts. Pronouns are the words used to replace nouns that are already known.

The two basic types of nouns are *proper nouns* and *common nouns*.

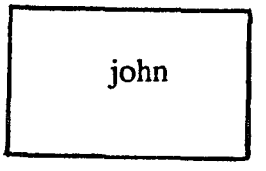
5.3.1.1.1. Proper nouns

Proper nouns name specific persons, places, things, or concepts. Examples of proper nouns are *Ronald-Reagan, Baton-Rouge, LSU, table24* and *book51*. Since proper nouns represent the names of specific things, real or imaginary, they correspond to the names of *entities* in Entity-Relationship approach as shown in figure 5.1.(a).

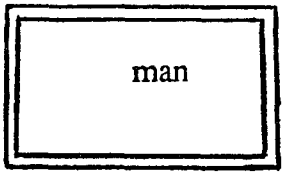
5.3.1.1.2. Common nouns

Common nouns name general classes or categories of persons, places, things, or concepts. Examples of common nouns include *man, table, book,* and *love*. Note that

common nouns can represent both abstract and concrete classes of things. Since common nouns are the names of general classes of things, they correspond to the names of *entity sets* as shown in figure 5.1.(b).



(a) ER-diagram for a proper noun.



(b) ER-diagram for a common noun.

Figure 5.1. Representation of nouns

5.3.1.2. Verbs

Verbs are used to describe an action, being, or a state of existence. Generally verbs establish a relationship between the main nouns in a sentence. Examples of verbs include *support, keep,* and *is.*

There are three types of verbs called: transitive, intransitive, and linking verbs.

5.3.1.2.1. Transitive Verbs

A transitive verb is one that requires a direct object, with or without an indirect object, to complete its meaning. It cannot describe the action performed by the subject without mentioning the object upon which the action is to be performed. Consider the following simple sentences that involve transitive verbs without an indirect object:

Students *take* courses.

John *attends* LSU.

Since transitive verbs relate subject and object(s) in the sentence, they correspond naturally with relationships in Entity-Relationship approach. A transitive verb may refer to either a specific relationship or a class of relationships (represented by a relationship set) depending on what is being described. If the transitive verb relates two common nouns as in 'Students take courses', it represents to a *relationship set* as shown in figure 5.2.(a), because it does not describe a specific instance of a relationship. Instead, it describes a set of relationships.

Consider the sentence 'John takes database'. Here the transitive verb 'takes' relates two proper nouns 'John' and 'database'. In this case, the transitive verb refers to a specific instance of the 'take' relationship such as 'take201'. Therefore, it corresponds to a *relationship* as in figure 5.2(b).

The sentence 'John takes courses' relates a proper noun with a common noun. This sentence refers to the set of all the specific instances in which John takes courses as shown in figure 5.2.(c). If this set is called take', then it becomes a *subset of the relationship set*, take, in figure 5.2.(a).

(a) Students take courses

(b) John takes database

(c) John takes courses

Figure 5.2. Representation of transitive verbs

Our discussion of transitive verbs so far dealt with only a subject and a direct object. Transitive verbs may also include an indirect object. An indirect object answers the questions 'to whom or what ?' or 'for whom or what ?'. It is normally the person or thing that receives the direct object. For example, in the sentence:

John gave Mary the printout

Mary is the indirect object to whom the direct object, the printout, is given by the subject John. While a sentence without an indirect object is represented by a *binary relationship* between the subject and the direct object, the sentence that includes the indirect object is represented by a *ternary relationship* among the subject, the direct object, and the indirect object as shown in figure 5.3.

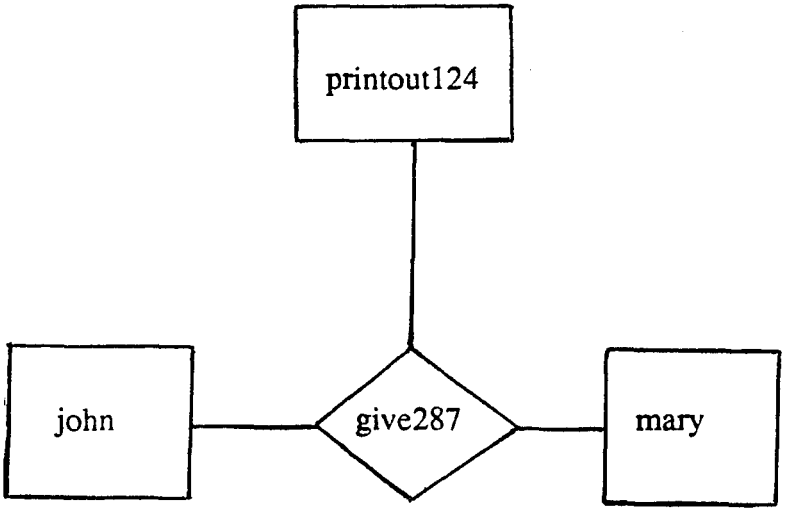
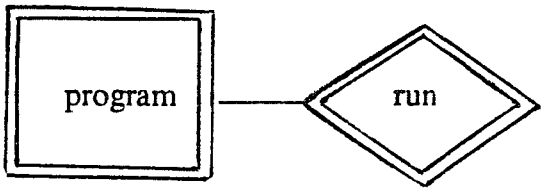


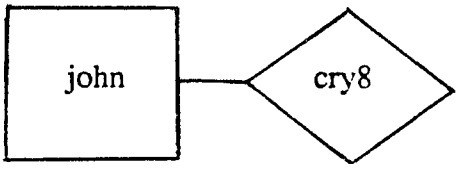
Figure 5.3. A transitive verb with an indirect object

5.3.1.2.2. Intransitive Verbs

An intransitive verb is a verb that does not require an object to complete its meaning. It is able to make a full assertion about the subject without assistance. The following are the two example sentences that have an intransitive verb.



(a) Programs *run.*



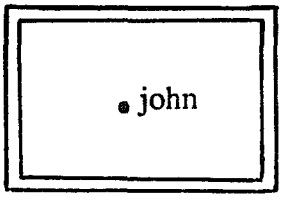
(b) John *cried.*

Figure 5.4. Representation of intransitive verbs

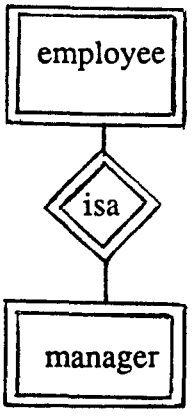
Intransitive verbs correspond to relationships that involve only one entity. If subject is a common noun, then the intransitive verb is treated as a *relationship set* as in figure 5.4.(a). Otherwise, it is treated as a *relationship* as in figure 5.4.(b).

5.3.1.2.3. Linking Verbs

A verb that functions primarily to link the subject to another noun or a modifier is called a linking verb or a copulative verb. The most common form of a linking verb is *be*. Consider the following examples of the verb 'be':



(a) John *became* a programmer.



(b) Managers *are* employees.

Figure 5.5. Representation of linking verbs

Here, the linking verb 'be' serves the useful function of indicating either *(1) members of a set or (2) subset of a set.* The sentence 'John became a programmer' indicates that an entity called John became a member of the entity set programmer. Therefore, if the linking verb 'be' relates a proper noun with a common noun, then it corresponds to the *membership of a specific entity in an entity set* as shown in figure 5.5.(a).

If the linking verb relates two common nouns, as in 'Managers are employees', then it indicates that one set is a subset of another. In this case, the linking verb corresponds to *an entity set being a subset of another entity set* as shown in figure 5.5.(b).

5.3.1.3. Articles

We noted in section 5.3.1.1.1. that individual entities are denoted by proper nouns. Articles provide another way to refer individual entities in an entity set when they modify common nouns. There are two kinds of articles, *indefinite* and *definite*.

Indefinite articles 'a' and 'an' *denote an unspecified entity in an entity set* as in the following example, *a* table:

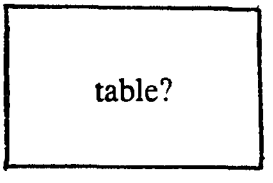


Figure 5.6. Representation of an indefinite article

Here, the indefinite article 'a' makes the noun phrase refer to an unspecified table, instead of a specific table. We assume that the entity referred is one of the members of the entity set 'table' without exactly specifying which table is being referred.

The definite article 'the' always *denotes a specific entity in an entity set* as in the following example, *the* table:

Figure 5.7. Representation of a definite article

The definite article 'the' modifies the common noun 'table' to refer to a specific table such as 'table24'. Therefore, we designate a specific entity in the entity set 'table'. Whenever a common noun is modified by the definite article 'the', it actually refers to a specific proper noun.

5.3.1.4. Adjectives

If the function of a word is to modify a noun or a pronoun then it is called an adjective. Adjectives are normally used to describe the qualities of a noun or a pronoun. The following noun phrases provide the examples of an adjective:

the *red color* car

the *6ft. long* table

The above examples indicate that the description of adjectives of a noun resembles the attribute description of an entity. In these examples, the noun is modified by two adjectives. The immediate adjective that modifies the noun corresponds to an *attribute* of the entity that represents the noun. The adjective that modifies the noun and its immediate adjective corresponds to the *value* of an attribute of an entity. For example, in the noun phrase 'the red color car', the adjective 'red' refers to the value of the attribute corresponding to the adjective 'color' of the entity 'car'.

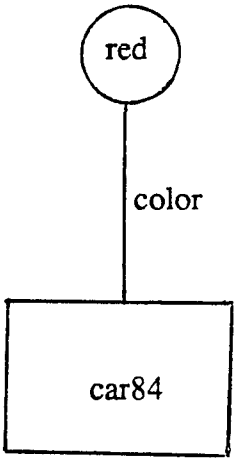


Figure 5.8. Representation of adjectives

Many times a noun may be modified by only a single adjective. In that case, do we treat the single adjective as an attribute or as a value ? The answer depends on the type of sentence that one is dealing with. While a single adjective in an *interrogative* sentence corresponds to an attribute, it corresponds to a value of an implicit attribute in a *declarative* sentence. Consider the following interrogative sentences:

What is the *color* of the car ?

What is the *length* of the table ?

These questions are inquiring the details of a quality of the noun. Therefore, the adjectives in these questions correspond to the attributes of an entity. Now consider the answers to the above questions:

It is a *red* car.

It is a *6ft.* table.

The adjectives in these answers correspond to the values of the attributes corresponding to the adjectives mentioned in the questions.

Aggregate operators such as 'percentage', 'average', and 'sum' are treated as attributes because they take on values. The following is one such example.

The building is 50 percent full.

Adjectives may also describe a *gerund.* Gerunds are the noun form of a verb. Therefore, the description of adjectives of a gerund naturally corresponds to the attribute description of a relationship in Entity-Relationship approach. Following is one such example:

Very fast running

Here, the gerund 'running' represents the relationship 'run' and 'fast' and 'very' correspond to an *attribute and its value of the relationship* 'run' as shown in figure 5.9. Note that the adjectives of a gerund are treated equivalent to the adverbs of the verb root form of the gerund (see section 3.5.1.5.)

5.3.1.5. Adverbs

If the function of a word is to modify a verb, then it is called an adverb. Adverbs describe the qualities of a verb as in the following sentences:

John ran *very fast*.

The water boiled *very hot*.

The description of an adverb resembles the description of an attribute of a relationship. Unlike in the case of adjectives, the immediate adverb of the modified verb corresponds to the *value of an attribute*, the *attribute* being the adverb that modifies the verb and its immediate adverb. In the example, 'John ran very fast', 'very' is the value of the attribute 'fast' of the relationship 'ran' as shown in figure 5.9.

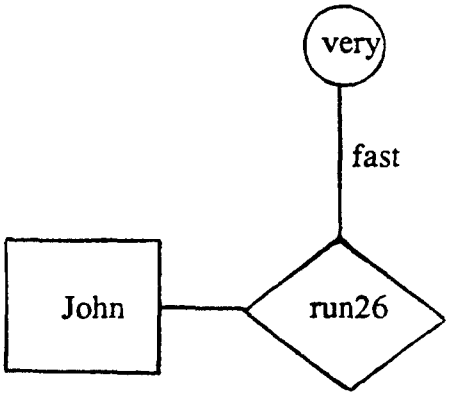


Figure 5.9. Representation of adverbs

In quite a few situations, the verb may be modified by a single adverb. A single adverb modifying a verb always corresponds to a *relationship attribute*. Consider the following cases of declarative and interrogative sentences.

John ran *fast.*

Did John run *fast* ?

In both the cases, the adverb represents a relationship attribute.

5.3.1.6. Prepositions

Preposition is a word when combined with another noun phrase forms a prepositional phrase. A prepositional phrase modifies a noun or a pronoun to express spatial or temporal relationships. Spatial relationships may indicate location (at, in, on, under, over, of, with, beside, among, by, between, through) or direction (to, into, across, toward, against, for) of the noun being described. Temporal relationships indicate time (before, after, during, until, since).

Since a prepositional phrase describes a relationship (spatial or a temporal relationship) between two or more nouns, it may be represented in Entity-Relationship approach by a *relationship* between two or more entities. Consider the example prepositions:

the table *in* the room

the mall *across* the street

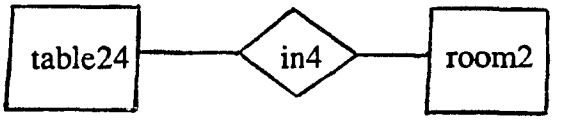


Figure 5.10. ER-diagram for the prepositional phrase 'the table in the room'

The first example is represented in figure 5.10. Our decision to represent prepositional phrases by relationships is justified because most prepositional phrases can be converted into a relative clause. For instance, the above examples can be expressed as the following relative clauses:

the table which is *in* the room

the mall which is *across* the street

In order to represent sentences that contain prepositional phrases, we will represent a noun phrase containing a prepositional phrase by a *high level entity*. The example sentence 'John owns the table in the room' is represented by the ER-diagram in figure 5.11.

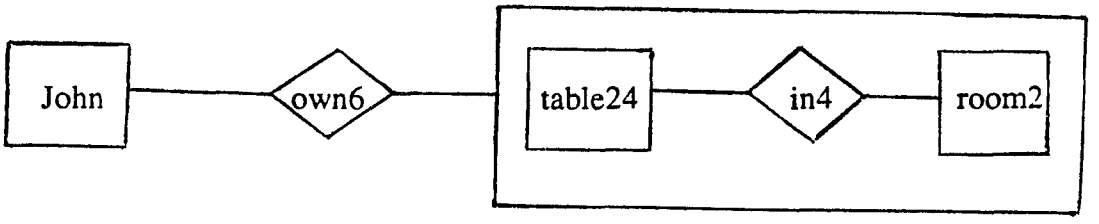


Figure 5.11. ER-diagram for the sentence 'John owns the table in the room'

5.3.2. A Formal Grammar

Syntactic approaches to natural language processing are based on a formal grammar such as a context-free grammar. The grammar of a language is, basically, a set of rules for constructing sentences from words and phrases. The grammar determines

what sentences are legal in the language. A sentence is a string of words, each having a different function in the sentence. These functions are defined by the categories, or parts of speech, into which words are classified: nouns, verbs, articles, adjectives, adverbs, and prepositions.

The basic syntactic form of most English sentences is represented by the following context-free grammar:

S	→	NP VP
NP	→	ADJ N PP
NP	→	ADJ N
NP	→	N PP
NP	→	N
NP	→	ART ADJ CN PP
NP	→	ART ADJ CN
NP	→	ART CN PP
NP	→	ART CN
N	→	CN
N	→	PN
ART	→	IART
ART	→	DART
VP	→	V NP
VP	→	V ADV
VP	→	V PP

VP	→	V
V	→	VI
V	→	VT NP
V	→	VL NP
PP	→	PREP NP
CN	→	man, table, room, book, course
PN	→	john, mary, bill, table24, book51
DART	→	the
IART	→	a, an
ADJ	→	red, color, long, 3 ft.
ADV	→	fast, slow, very
PREP	→	with, in, across, near
VI	→	cry, run
VT	→	support, keep, own, tell, break
VL	→	is, become

where

S	sentence
NP	noun phrase
VP	verb phrase
N	noun
CN	common noun
PN	proper noun

ART	article
DART	definite article
IART	indefinite article
ADV	adjective
V	verb
VI	intransitive verb
VT	transitive verb
VL	linking verb
PP	prepositional phrase
PREP	preposition

5.3.3. Parse Trees

The most common form of syntactic analysis of a sentence is a parse tree. Consider the following example sentence:

John broke the window with a hammer.

When analyzed, this sentence yields the parse tree shown in figure 5.12. We observe that the parse tree allows one to decompose a sentence into different parts of speech, which are then translated into corresponding ER-primitives based on our discussion in the section 5.3.1. The corresponding ER-diagram for the parse tree in figure 5.12 is shown in figure 5.13.

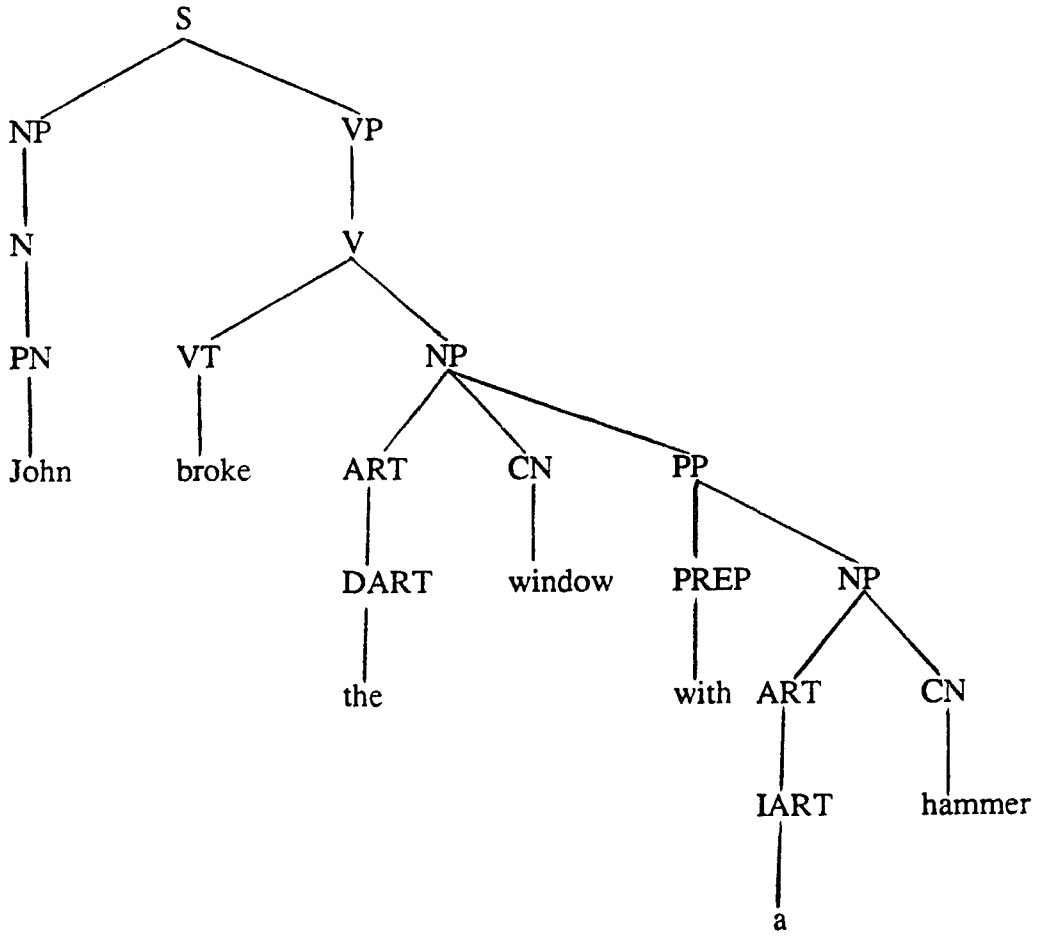


Figure 5.12. Parse tree for the sentence 'John broke the window with a hammer'

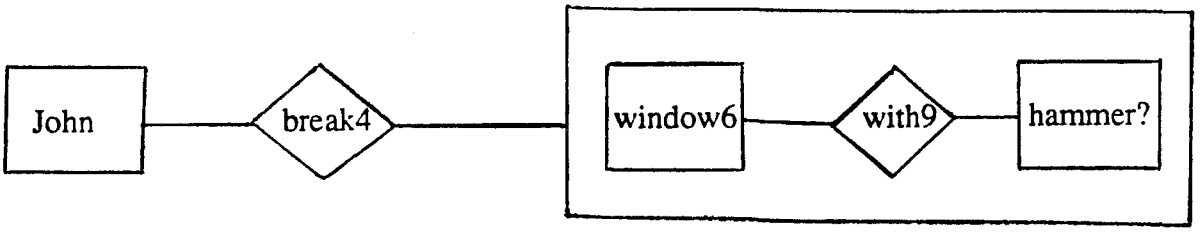


Figure 5.13. ER-diagram for the sentence 'John broke the window with a hammer'

5.4. Deep Structure

Although syntactic grammars are simple to define, they suffer the serious problem of permitting sentences that are syntactically correct but semantically meaningless. The grammar in section 5.3.2, for example, allows the meaningless sentence:

The hammer broke the John with a window.

Furthermore, it is difficult to handle sentence information such as voice (active or passive), tense (past, present, future), mood (declarative, interrogative, imperative), etc in syntactic approaches. In order to handle these issues, a sentence is transformed into its deep structure that lends itself to a more effective semantic analysis.

Linguists realized [Cho65, Fil68, KaF64] that one should consider the intentions behind the utterance of a sentence to understand its meaning. Consider the two sentences: 'It surprised John that Mary arrived on time' and 'That Mary arrived on time surprised John'. These two sentences are quite different on the surface. The first sentence contains a word, *it*, that the second sentence does not contain, and the order of the shared words is quite different between the two. Although they have apparently different surface structure, the two sentences have the same deep structure. Thus, two sentences with the same meaning will have the same deep structure.

Of the different approaches to deep structure of a sentence, Case Grammars [Bru75, Fil68, Win83] have attracted significant attention due to their simplicity in handling the semantic relationships between the parts of speech of a sentence we discussed in section 5.3. The purpose of this section is to show the relationship between Case Grammars and Entity-Relationship approach. The natural correspondence between them supports our contention that Entity-Relationship approach is suitable

for representing the deep structure of a sentence effectively.

5.4.1. Case Grammar

Traditionally, case forms are applied to nouns and pronouns to show the relationship of each word to the other words in the sentence. In a very influential paper, Fillmore [Fil68], however, argued and showed that the verb plays the central role in determining the relationships between the words in a sentence instead of a noun or a pronoun. Fillmore postulated that

> The sentence in its basic structure consists of a verb and one or more noun phrases, each associated with the verb in a particular case relationship [pp. 21, 1968].

Thus, Fillmore's case grammar begins with the verb which predicates something. All other elements in the sentence occupy semantic roles or cases in relation to the verb. Consider the sentence:

John broke the window with a hammer.

Here, the verb *break* describes the action, where John, the person performing the action is the *agent*; hammer, the object used in the action is called the *instrument*; and window is the *recipient* of the action.

According to Fillmore, a sentence is made up of

> the *proposition*, a tenseless set of relationships involving verbs and nouns (and embedded sentences, if there are any), separated from what might be called the *modality* constituent. This latter will include such modalities on the sentence as a whole as negation, tense, mood, and aspect [pp. 23, 1968].

Therefore, the deep structure of a sentence, S, is divided into its two components: the proposition, P, and the modality, M. In the next two sections, we describe the

meaning of the proposition and the modality components of a sentence and their representation in Entity-Relationship approach.

5.4.2. Proposition

The proposition in a sentence consists of a verb and the various cases related to the verb filled by different nouns. Thus, the verb provides the organization of a proposition. In our example sentence 'John broke the window with a hammer', the verb 'break' has the three cases: agent, object, and instrument filled by the nouns: John, window, and hammer respectively.

The proposition of the sentence 'John broke the window with a hammer' is represented by the ER-diagram in figure 5.14.

5.4.2.1. Mandatory Cases

Fillmore proposed a list of cases for each verb. Consider the following four sentences:

1. John broke the window.

2. A hammer broke the window.

3. John broke the window with a hammer.

4. The window broke.

Although the apparent surface structure of these four sentences is different, all of them seem to describe the same event about 'breaking' something. By introducing deep cases, we can account for their meaning. In the sentences 1 and 4 John is the agent, or the individual responsible for the action, hammer is the instrument, or the

object involved in the action; and window is the object of the action.

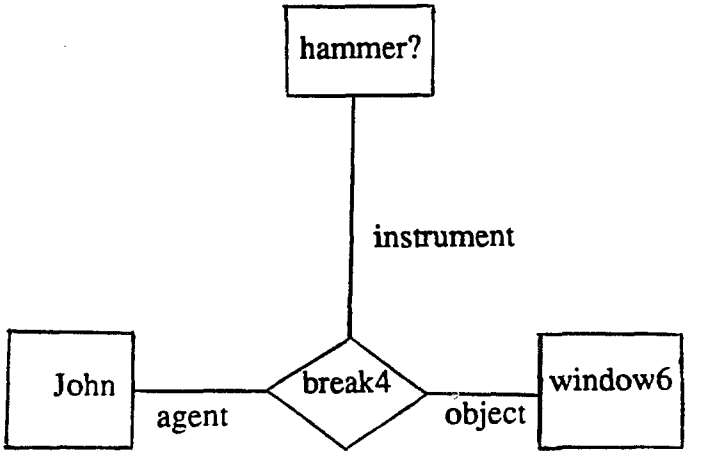


Figure 5.14. An ER-diagram representing the deep cases of the sentence 'John broke the window with a hammer'

The difference between the above sentences can be explained by the type of cases that the verb is associated with. Cases in a verb can be classified into mandatory, optional, and forbidden cases. For example, in the verb 'break' the object case is mandatory without which a sentence is not complete. This is the reason the object 'window' appears in all of the example sentences.

5.4.2.2. Optional Cases

Not all cases associated with a given verb need to be present always. Some cases of the verb can be optional. The presence of an optional case, although not necessary for a the completion of the meaning of a sentence, provides additional information.

For example, the verb 'break' has the agent as an optional case. This explains the reason why the above sentences 2 and 4 are permissible.

5.4.2.3. Forbidden Cases

It is possible that some cases of a verb to be forbidden. Forbidden cases of the verb in a sentence are unfilled. For example, the case 'direction' in the verb 'break' is forbidden. Note that this case is not filled in any of the above example sentences.

Since cases correspond to roles of a relationship in Entity-Relationship approach, the above three types of cases are represented by the mandatory, optional, and forbidden roles of a relationship.

The common cases normally used in natural language analysis include: agent, object, instrument, location and direction. These cases are assigned to nouns in a sentence based on the following prepositions associated with them:

agent: by (or none)

object: none

instrument: with

location: on, in, under, at

direction: to, for, against.

Consider our example sentence: 'John broke the window with a hammer'. Both the noun phrases 'John', and 'the window' are not associated with any preposition. In such cases, the first noun phrase is treated as the agent, while the second noun phrase is treated as the object. Since the noun phrase 'a hammer' is preceded by the preposi-

tion 'by', it is identified as the 'instrument'.

5.4.3. Modality

Certain types of information about a sentence can be separated from its proposition as the modality component. The types of information included in the modality component are the tense, the mood, the voice, and the essence.

5.4.3.1. Tense

The tense of a sentence deals with the time orientation of the sentence. It indicates when the proposition mentioned in the sentence occurred whether it is past, present, or future. The following are the example sentences:

John broke the window with a hammer. (past tense)

John is breaking the window with a hammer. (present tense)

John will break the window with a hammer. (future tense)

The tense information of a sentence is represented by the attribute 'tense' of the entity that represents the sentence as shown in figure 5.15.

5.4.3.2. Voice

Voice indicates the relation of the subject to the action of the verb. When the verb is in the active voice, the subject acts, when it is in passive voice, it is acted upon. Consider the following example sentences:

John broke the window with a hammer. (active voice)

The window was broken by John with a hammer. (passive voice)

The figure 5.15 shows the representation of voice information of a sentence by the attribute 'voice' of the entity that represents the sentence.

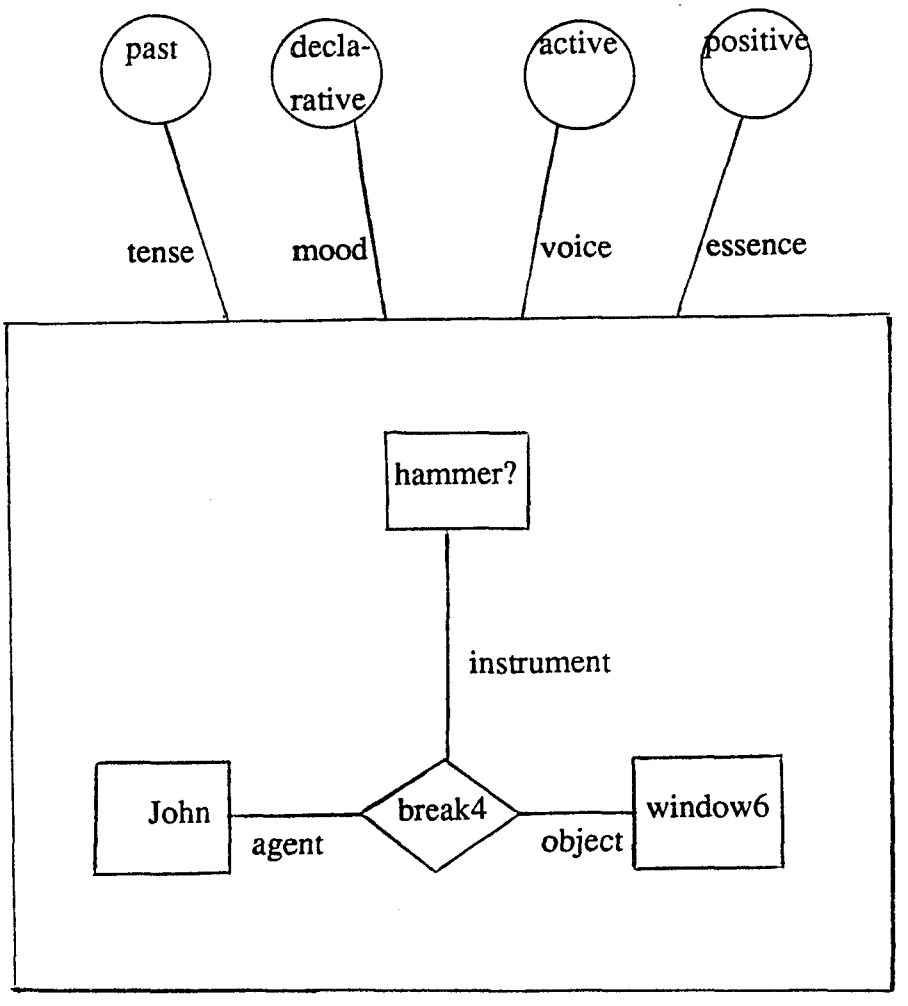


Figure 5.15. An ER-diagram representing propositional and modality components of a sentence

5.4.3.3. Essence

Essence of a sentence indicates whether the proposition dealt in the sentence is a positive statement or a negative statement. Consider the example sentences:

John broke the window with a hammer. (positive essence)

John did not break the window with a hammer. (negative essence)

The essence information of a sentence is represented by the attribute 'essence' of the entity that represents the sentence as shown in figure 5.15.

5.4.3.4. Mood

The mood of a sentence indicates whether the sentence intends (1) to make a statement (declarative), (2) to ask a question (interrogative mood), (3) to give a command (imperative), or (4) to express feelings (exclamatory). The following are the example sentences:

John broke the window with a hammer. (declarative mood)

Did John break the window with a hammer ? (interrogative mood)

Break the window with a hammer. (imperative mood)

For heavens sake, why did John break the window ! (exclamatory mood)

The mood of a sentence is represented by an attribute 'mood' of the entity representing the proposition of the sentence which may assume one of the values: declarative, interrogative, imperative, or exclamatory as in figure 5.15.

5.4.4. Surface and Deep ER-Diagrams

It may be noted that the figures 5.13 and 5.15 represent the same sentence 'John broke the window with a hammer'. However, there are important differences between the two figures, which are attributed to two different types of analysis performed on the same sentence, one at the surface level and the other at a more deeper level.

First, the verb plays the central role in structuring the ER-diagram in figure 5.15. On the other hand, the ER-diagram in figure 5.13 is organized by the syntactic phrases, such as noun phrase, NP, and verb phrase, VP, in a sentence.

Second, in figure 5.13, the role information is missing in the ER-diagram, while it is made use very effectively in figure 5.15. Role information is derived from the prepositions present in the sentence. Therefore, prepositions are absent in the figure 5.15, which are replaced by their corresponding roles. On the other hand, in figure 5.13, a preposition is treated as a relationship and, therefore, it is represented in the ER-diagram itself. The syntactic and the semantic (i.e. case grammar) grammars presented in sections 5.3.2 and 5.4 work differently on the same parts of speech described in section 5.3.1, and yield structures represented by different ER-diagrams.

Third, the figure 5.15 incorporates useful modality information which is absent in figure 5.13. Therefore, we find that deep ER-diagrams capture more semantics than a surface ER-diagram.

It may be noted that the surface ER-diagram in figure 5.13 resembles traditional ER-diagrams, while the deep ER-diagram based on the notation introduced in chapter 2 differs from them by adding more semantic details.

5.4.5. Correspondence with Entity-Relationship Approach

Our discussion in the last two sections 5.4.2 and 5.4.3 indicates that there is a natural correspondence between Case Grammar and the Entity-Relationship approach followed in this dissertation. Every sentence, S, is represented by an entity, E. The proposition and modality components, P and M, of the sentence, S, correspond to

internal and external descriptions, $E_{internal}$ and $E_{external}$, of the entity, E, shown in figure 5.16.

The internal description, $E_{internal}$, of the entity, E, contains its component entities, E_1, E_2, ..., E_n, that correspond to the noun phrases, NP_1, NP_2, ..., NP_n, of the sentence, S. While the component entities, E_1, E_2, ..., E_n, of the entity, E, are organized by the relationship, R, the noun phrases, NP_1, NP_2, ..., NP_n, of the sentence, S, are organized by the verb, V. Note that the relationship, R, and the verb, V, play the central role in organizing the entity, E, and the sentence, S, respectively. Therefore, the relationship, R, in entity, E, corresponds to the verb, V, in the sentence, S.

Each verb, V, in Case Grammar is associated with a set of cases c_1, c_2, ..., c_n filled by noun phrases NP_1, NP_2, ..., NP_n. Correspondingly, in Entity-Relationship approach, the roles r_1, r_2, ..., r_n associated with the relationship, R, representing the verb, V, are performed by the entities, E_1, E_2, ..., E_n. Therefore, for each case, c_i, of the verb, V, there is a corresponding role, r_i, in the relationship, R. Note that the case assigned to a given noun phrase is determined by the preposition that links the noun phrase with the verb as discussed in section 5.4.2.3.

The modality component, M, of a sentence, S, corresponds to the external description, $E_{external}$, of the entity, E, representing the sentence, S. Each modality, m_i, of the sentence, S, has a corresponding attribute, a_i, of the entity, E. The attribute, a_i, of the entity, E, assumes the same value, V_i, that the modality, m_i, of the sentence, S, assumes.

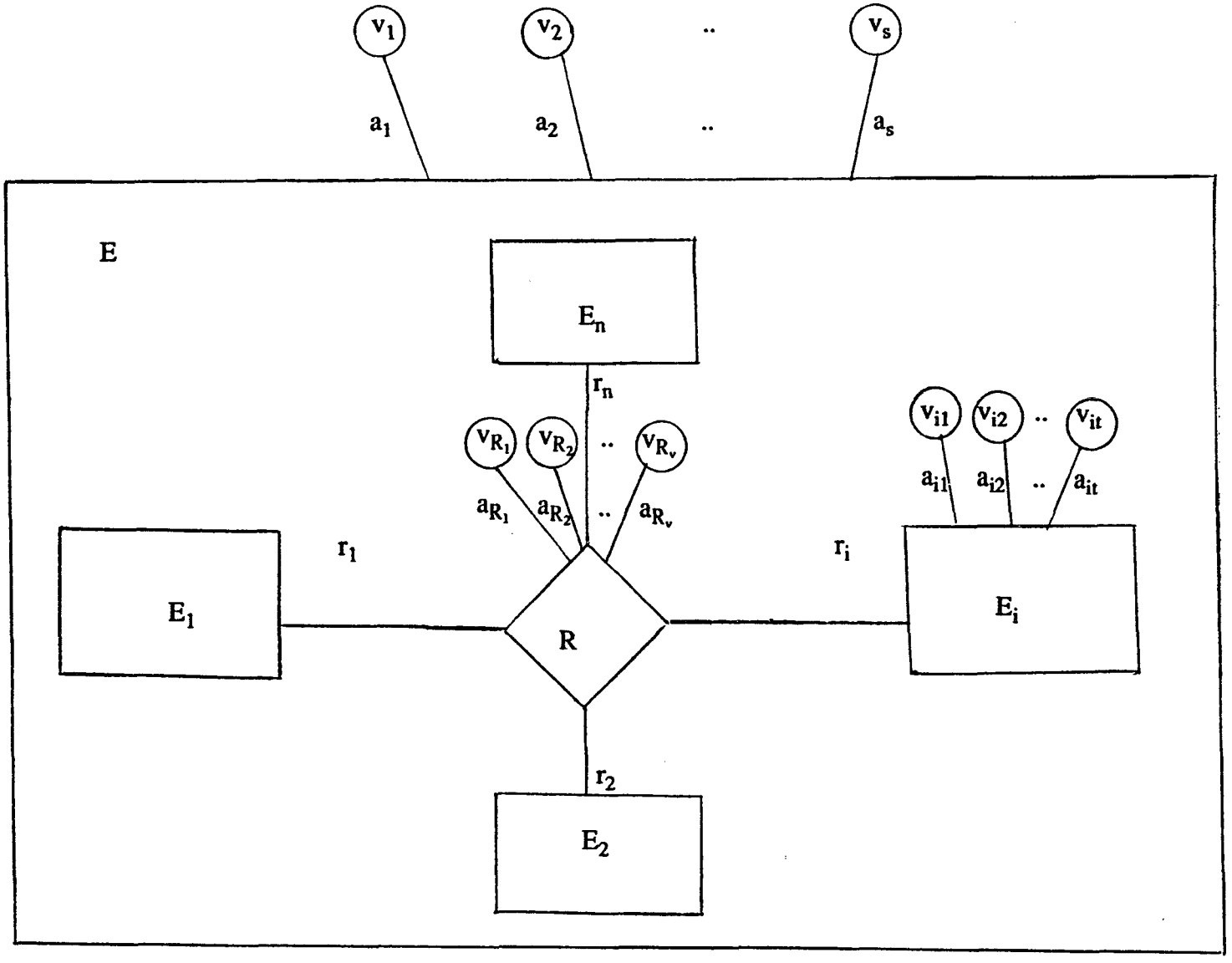


Figure 5.16. Representation of Case Grammar by an ER-diagram

Adjectives are associated with noun phrases in a sentence. Each adjective, 'adj_{ij}', of a noun phrase, NP_i, in sentence, S, corresponds to the attribute, a_{ij} of the component entity, E_i, of the entity, E.

An adverb in a sentence describes the verb. Each adverb, adv_i, of the verb, V, in the sentence, S, corresponds to the attribute, a_{R_i}, of the relationship, R, in the entity, E.

The following is a summary of the correspondence between Case Grammar and Entity-Relationship approach:

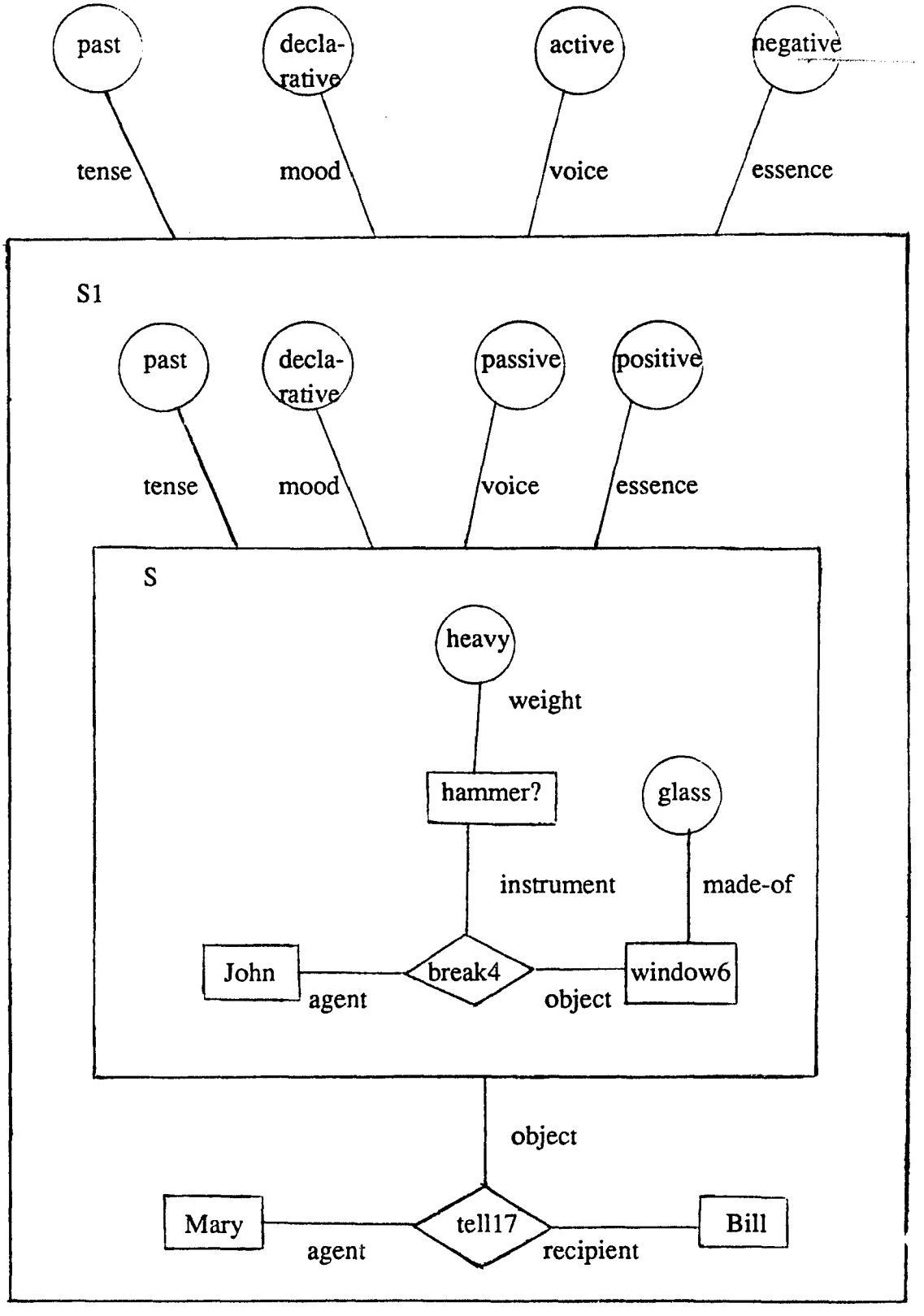
Case Grammar	**Entity-Relationship Approach**
S	E
P	$E_{internal}$
M	$E_{external}$
V	R
NP_1	E_1
NP_2	E_2
..	..
NP_n	E_n
c_1	r_1
c_2	r_2
..	..
c_n	r_n
m_1	a_1

m_2	a_2
..	..
m_s	a_s
adj_{i1}	a_{i1}
adj_{i2}	a_{i2}
..	..
adj_{it}	a_{it}
adv_1	a_{R_1}
adv_2	a_{R_2}
..	..
adv_v	a_{R_v}

where the symbols have the meaning defined in this section.

The major reason for the elegant correspondence between Case Grammars and the Entity-Relationship approach used in this dissertation is the similarity between the nature of relationships and verbs in entities and sentences respectively. While relationships provide organization to form meaningful wholes from the parts, verbs structure the phrases to form a meaningful sentence.

The holistic notation proposed in Chapter II for Entity-Relationship approach is also useful for handling embedded sentences. Consider, for example, the sentence, S, 'The glass window was broken by John with a heavy hammer' being embedded in the sentence, S_1: 'Mary did not tell Bill that the glass window was broken by John with a heavy hammer'. It is represented by the ER-diagram in figure 5.17.



5.17. An ER-diagram for an embedded sentence

The figure 5.17 can be translated into a first-order logic system, a production system, or a frame-based system using the translation rules proposed in chapters 3 and 4. The following, for example, is a frame-based representation of the ER-diagram in figure 5.17:

```
*(john
      (#agent          (value      (break4)))
      (w-entities      (value      (S)))
)

*(window6
      (@made-of        (value      (glass)))
      (#object         (value      (break4)))
      (w-entities      (value      (S)))
)

*(hammer?
      (@weight         (value      (heavy)))
      (#instrument     (value      (break4))
      (w-entities      (value      (S))) )

*(S
      (@tense          (value      (past)))
      (@mood           (value      (declarative)))
      (@voice          (value      (passive)))
      (@essence        (value      (positive)))
      (#object         (value      (tell)))
      (w-entities      (value      (S1)))
      (p-entities      (value      (john, window6, hammer?)))
)

*(mary
      (#agent          (value      (tell17))
      (w-entities      (value      (S1)))
)

*(bill
      (#recipient      (value      (tell17))
      (w-entities      (value      (S1)))
)
```

```
*(S1
      (@tense          (value      (past)))
      (@mood           (value      (declarative)))
      (@voice          (value      (active)))
      (@essence        (value      (negative)))
      (p-entities      (value      (mary, S, bill)))
      (o-relationships (value      (tell17)))
)

&(break4
      (o-entities      (value      (S)))
      (#agent          (value      (john)))
      (#object         (value      (window6)))
      (#instrument     (value      (hammer?)))
)

&(tell17
      (o-entities      (value      (S1)))
      (#agent          (value      (mary)))
      (#object         (value      (S)))
      (#recipient  (value      (bill)))
)
```

This chapter provided a basis for analyzing the descriptions in natural language using Entity-Relationship approach. A user front-end of a knowledge base system can be used to convert English descriptions into ER-diagrams. The ER-diagrams obtained through this analysis can be translated into first-order logic, productions, or frames using the discussion in chapters 3 and 4. The symbolic data structures obtained from the translation process are then driven by a knowledge base system capable of maintaining the first-order system, the production system, or the frame-based system implemented.

CHAPTER VI

Conclusions

6.1. Introduction

This chapter presents a summary of the major contributions of the work described in this dissertation to knowledge base systems and Entity-Relationship approach. It also suggests the directions for future research.

6.2. Contributions to Knowledge Base Systems

The first major contribution of this dissertation is that it provided a unified framework for knowledge base systems using Entity-Relationship approach. First-order logic systems, production systems, and frame-based systems are derived from Entity-Relationship approach by using the translation rules provided in chapters 3 and 4. Thus, knowledge base systems can be viewed through a window provided by Entity-Relationship approach.

The second contribution is that it divided the knowledge base development into two separate manageable phases: one for the specification and another for the implementation. Most often the phases in the development cycle of a knowledge base overlap. Therefore, the approach proposed here contributes towards the goal of improving the software development process.

The third contribution is that it emphasized the need for a holistic representation to capture real-world perceptions. Perception is known to be a holistic phenomenon [ASH87, BDL79, Cod70, Zus70], where things are perceived as wholes that stand out

of their general background. A whole emerges out of its general background due to the organization among its parts. In knowledge representation, a concept is usually treated as a sum of its parts by ignoring the organizational aspects that make the parts to emerge as a single whole. It was observed by Gestalt Psychologists [BDL79, Koh47, KuP81] that a whole is usually different form its parts. Chapter 2 showed the explicit representation of the whole, the parts, and the process of integration of the parts that forms the whole using the semantic primitives provided by Entity-Relationship approach. The dissertation also used the same holistic notation through out to demonstrate the usefulness of a holistic representation in specification, and in choosing the data structures for the implementation, and in natural language analysis.

The fourth contribution is that it incorporated an underlying theory of the structure and the function of things into the proposed holistic representation. The distinction between the structure and the function helps the development of knowledge bases for physical systems.

In representations, hierarchies are normally formed to permit inheritance of properties. Normally, hierarchies treat different types of descriptions uniformly as properties. Chapter 2 showed that different types of descriptions such as parts, wholes, properties, roles, and organizing relationships must be distinguished. Therefore, hierarchies of things must include properties as well as other descriptions. Thus, specific classes in a hierarchy inherit from their general classes not only properties but also other descriptions as well.

The sixth contribution is that it pointed out the inappropriateness of the inheritance of properties from parts to the whole in a *hasa* hierarchy. The observation that a whole is different from its parts precludes the automatic inheritance of properties from parts to wholes. What is inherited by a whole in a *hasa* hierarchy is the parts of its parts instead of the properties of its parts.

6.3. Contributions to Entity-Relationship Approach

The first contribution to Entity-Relationship approach is that it is extended to knowledge base design. Traditionally, Entity-Relationship approach is widely used in data base design. The major benefit of this extension is that Entity-Relationship approach can now be used as a unifying framework for the design of both data base and knowledge base systems. Thus, it provides a conceptual framework under which a spectrum of data and knowledge base systems can be developed.

Although other attempts are underway to use Entity-Relationship approach in knowledge base systems [FeF85, Laz87, SeS85], it is not yet clear about its exact role in the development cycle of a knowledge base system. The contribution of this dissertation is that it advocates the use of Entity-Relationship approach in the initial stages of software development.

The third contribution is that it added the semantic primitive, *role*, to the core set of primitives: *entities, relationships, attributes,* and *values* in Entity-Relationship approach. The introduction of the primitive, *role*, enriched the semantics of the model in capturing real-world perceptions.

The fourth contribution is that it developed a basis for using Entity-Relationship approach for the specification of real-world perceptions. Since an approach for the specification of real-world perceptions must provide appropriate semantic primitives, it offered a justification for the semantic primitives proposed in Entity-Relationship approach by considering the fundamental issues in perception. It also developed a notation that allows Entity-Relationship approach to be used as a holistic representation.

Entity-Relationship approach was criticized for the lack of providing structure to represent complex descriptions [BPR88]. The holistic notation proposed in this dissertation provides a solution to this criticism by providing a notation to describe complex entities. The notation allows entities to be represented in isolation or as part of other entities. It also allows the representation of the integration of the component entities that makes an entity to emerge.

The sixth contribution is that it provided a framework for using Entity-Relationship approach to support a natural language front-end of a knowledge base system by analyzing the correspondence of surface and deep structures of sentence with ER-diagrams. The work initiated by Chen on surface structure analysis is extended to different types of verbs, articles, and prepositions. In addition, it established the correspondence between Entity-Relationship approach and Case Grammars to represent the deep structure of a sentence.

6.4. Future Directions

The following are some directions for the future research of the work described in this dissertation:

First, an attempt may be made to integrate heterogeneous knowledge bases under a view provided by Entity-Relationship approach. It is known that providing a single enterprise view of N systems to M different user interfaces requires only N+M inter-connections instead of NM [Che77].

Second, the ideas presented in this dissertation may be applied to object-oriented system design [Boo86, Ren82]. Particularly, the external and internal descriptions of entities correspond to the structure of packages. Since the approach presented here is centered around objects and their interactions, it is a suitable candidate for object-oriented design.

Third, the parts and wholes analysis in this dissertation may be used to refine the inheritance in hierarchies. Inheritance hierarchies are an active area of research [EtR83, Tou86]. Our analysis adds a new dimension to the complexity of the problem.

Bibliography

[Abr74] J. R. Abrial, Data Semantics, *Data Base Management (J.W. Klimbie and K.L. Koffeman, Eds.)*, Amsterdam, Netherlands, 1974, 1-59.

[AsS88] R. N. Aslin and L. B. Smith, Perceptual Development, *Annual Review of Psychology (M.R. Rosenzweig and L.W. Porter Eds.) 39* (1988), 435-473, Annual Reviews Inc.

[ASH87] R. L. Atkinson, E. E. Smith and E. R. Hilgard, *Introduction to Psychology*, Harcourt, Brace, Jovanovich, Publishers, San Dieago, California, 1987.

[BaF81] A. Barr and E. Feigenbaum, *The Handbook of Artificial Intelligence, Volumes 1-2*, William Kaufman, Palo Alto, California, 1981.

[Bar32] F. C. Bartlett, *Remembering*, Cambridge University Press, Cambridge, England, 1932.

[BaS80] C. Batini and G. Santucci, Top-Down Design in the Entity-Relationship Model, *Entity-Relationship Approach to Systems Analysis and Modeling (P.P.S. Chen, Eds.)*, Amsterdam, Netherlands, 1980, 103-119.

[BiG86] L. Bic and J. P. Gilbert, Learning from AI: New Trends in Database Technology, *IEEE Computer*, 1986, 44-54.

[BPR88] M. R. Blaha, W. J. Premerlani and J. E. Rumbaugh, Relational Database Design Using an Object-Oriented Methodology, *Communications of the ACM 31*, 4 (April 1988), 414-427.

[Blo76] C. M. Bloomer, *Principles of Visual Perception*, Van Nostrand Reinhold Company, New York, New York, 1976.

[BoW77] D. G. Bobrow and T. Winograd, An Overview of KRL, a Knowledge Representation Language, *Cognitive Science 1*, 1 (1977), 263-286.

[Boo86] G. Booch, Object-Oriented Development, *IEEE Transactions on Software Engineering SE-12*, 2 (February 1986), 211-221.

[Bor80] S. A. Borkin, *Data Models: A Semantic Approach for Database Systems*, MIT Press, Cambridge, Massachusetts, 1980.

[BDL79] L. E. Bourne, R. L. Dominowski and E. F. Loftus, *Cognitive Processes*, Prentice-Hall, Inc , Englewood Cliffs, New Jersey, 1979.

[Bra78] R. J. Brachman, KL-ONE Reference Manual, *BBN Report 3848*, Cambridge, MA, 1978.

[Bra79] R. J. Brachman, On the Epistemological Status of Semantic Networks, *Associative Networks (N.V. Findler Eds.)*, New York, New York, 1979.

[BrS80] R. J. Brachman and B. C. Smith, Special Issue on Knowledge Representation, *SIGART Newsletter*, 1980.

[BrL85] R. J. Brachman and H. J. Levesque, *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, California, 1985.

[BMS84] M. L. Brodie, J. Mylopoulos and J. W. Schmidt, *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer-Verlag, New York, New York, 1984.

[BrM86] M. L. Brodie and J. Mylopoulos, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, New York, New York, 1986.

[Bru75] B. Bruce, Case Systems for Natural Language, *Artificial Intelligence 6* (1975), 327-360.

[BrE86] G. Bruno and A. Elia, Extending the Entity Relationship Approach for Dynamic Modelling Purposes, *5th International Conference on Entity-Relationship Approach*, Dijon, France, November 17-19, 1986, 311-323.

[Bun83] A. Bundy, *The Computer Modelling of Mathematical Reasoning*, Academic Press, New York, New York, 1983.

[ChL73] C. L. Chang and R. C. T. Lee, *Symbolic Logic and Mathematical Theorem Proving*, Academic Press, New York, New York, 1973.

[Che76] P. P. S. Chen, The Entity-Relationship Model: Toward a Unified View of Data, *ACM Transactions On Database Systems 1*, 1 (March 1976), 9-36.

[Che77a] P. P. S. Chen, The Entity-Relationship Model - A Basis for the Enterprise View of Data, *AFIPS Conference Proceedings*, Montvale, New Jersey, 1977, 77-84 .

[Che77b] P. P. S. Chen, The Entity-Relationship Approach to Logical Data Base Design, *Q.E.D. Monograph Series* , Wellesly, Massachusetts, 1977.

[Che80] P. P. S. Chen, *Entity-Relationship Approach to Systems Analysis and Modeling*, North Holland, Amsterdam, Netherlands, 1980.

[Che81] P. P. S. Chen, *Entity-Relationship Approach to Information Modeling and Analysis*, North Holland, Amsterdam, Netherlands, 1981.

[Che83] P. P. S. Chen, English Sentence Structure and Entity-Relationship Diagrams, *Information Sciences 29* (May 1983), 127-149, Elsevier Science Publishers.

[ChL86] P. P. Chen and M. Li, The Lattice Structure of Entity Set, *5th International Conference on Entity-Relationship Approach*, Dijon, France, November 17-19, 1986, 311-323.

[Cho65] N. Chomsky, *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Massachusetts, 1965.

[Cod70] E. F. Codd, A Relational Model for Large Shared Data Banks, *Communications of the ACM 13*, 6 (June 1970), 377-387.

[Cod79] E. F. Codd, Extending the Database Relational Model to Capture More Meaning, *ACM Transactions on Database Systems 4*, 4 (1979), 397-434.

[Cop86] I. M. Copi, *Introduction to Logic*, MacMillan Publishing Company, New York, New York, 1986.

[DaB85] M. W. Daehler and D. Bukato, *Cognitive Development*, Alfred A. Knopf, New York, New York, 1985.

[Dah83] V. Dahl, Logic Programming as a Representation of Knowledge, *IEEE Computer 16*, 10 (1983), 106-113.

[DaK77] R. Davis and J. King, An Overview of Production Systems, *Machine Intelligence 8 (E. Elcock and D. Michie, Eds.)*, Chichester, England,

1977, 300-332.

[DJN83] C. G. Davis, S. Jajodia and P. A. Ng, *Entity-Relationship Approach to Software Engineering*, North Holland, Amsterdam, Netherlands, 1983.

[Dav85] R. E. Davis, Logic Programming and Prolog: A Tutorial, *IEEE Software*, September 1985, 53-62.

[DeK79] A. Deliyanni and R. A. Kowalski, Logic and Semantic Networks, *Communications of the ACM 22*, 3 (March 1979), 184-192.

[Dry81] H. L. Dryfus, From Micro-Worlds to Knowledge Representation: AI at an Impasse, *Mind Design (J. Hagueland, Eds.)*, Cambridge, Massachusetts, 1981, 161-204.

[ERC85] *The 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, October 28-30, 1985.

[EWH86] R. Elmarsi, J. Weeldreyer and A. Hevner, The Category Concept: An Extension to the Entity-Relationship Model, *Data & Knowledge Engineering 1* (1986), 75-116, North Holland.

[EtR83] D. W. Etherington and R. Reiter, On Inheritance Hierarchies with Exceptions, *Proceeding of AAAI-83*, Washington, DC, 1983, 104-108.

[Fah79] S. E. Fahlman, *NETL: A System for Representing and Using Real World Knowledge*, MIT Press, Cambridge, Massachusetts, 1979.

[Fai85] R. Fairly, *Software Engineering Concepts*, McGraw-Hill, New York, New York, 1985.

[FeF85] P. Feldman and G. Fitzgerald, Representing Rules Through Modelling Entity Behavior, *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, October 28-30, 1985, 189-198.

[FiK85] R. Fikes and T. Kehler, The Role of Frame-Based Representation in Reasoning, *Communications of the ACM 28*, 9 (September 1985), 904-920.

[Fil68] C. Fillmore, The Case for Case, *Universals in Linguistic Theory (E. Bach and R.T. Harms Eds.)*, New York, New York, 1968.

[For81] C. L. Forgy, OPS5 User's Manual, Tech. Report No: CMU, Carnegie Mellon University, Pittsburgh, PA, 1981.

[FoH78] D. J. Foss and D. T. Hakes, *Psycholinguistics: An Introduction to the Psychology of Language*, Prentice Hall, Englewood Cliffs, New Jersey, 1978.

[Fox79] M. S. Fox, On Inheritance in Knowledge Representation, *Proceedings of IJCAI-79*, Tokyo, Japan, 1979, 282-284.

[GeG85] M. R. Genesreth and M. L. Ginsberg, Logic Programming, *Communications of the ACM 28*, 9 (September 1985), 933-941.

[GeN87] M. R. Genesreth and N. J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Palo Alto, California, 1987.

[GBG87] R. M. Granovskaya, I. Y. Breznaya and A. N. Grigorieva, *Perception of Form & Form of Perception*, Lawrence Earlbaum Associates, Hillsdale,

New Jersey, 1987.

[HaM81] M. Hammer and D. McLeod, Database Description with SDM: A Semantic Database Model, *ACM Transactions on Database Systems 6*, 3 (1981), 351-386.

[Haw84] I. T. Hawryszkiewycz, *Database Analysis and Design*, Science Research Associates Associates, Chicago, Illinois, 1984.

[Hay74] P. J. Hayes, Some Problems and Non-Problems in Representation Theory, *Proceedings of AISB Summer Conference*, Sussex, England, 1974, 63-79.

[Hay77] P. J. Hayes, In Defence of Logic, *Proceedings of IJCAI-77*, Cambridge, Massachusetts, 1977, 559-565.

[Hay79] P. J. Hayes, The Logic of Frames, *Frame Conceptions and Text Understanding (D. Metzing, Eds.)*, Berlin, West Germany, 1979, 46-61.

[HaC83] P. J. Hayes and J. G. Carbonell, A Tutorial on Techniques and Applications for Natural Language Processing, Tech. Report No: CMU, Carnegie Mellon University, Pittsburgh, PA, 1983.

[Hay85] F. Hayes-Roth, Rule-Based Systems, *Communications of the ACM 28*, 9 (September 1985), 921-932.

[HeC85] J. P. Held and J. V. Carlis, Conceptual Data Modeling of an Expert System, *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, October 28-30, 1985, 182-188.

[Hof80] D. R. Hofstadter, *Godel, Escher, Bach: An Eternal Golden Braid*, Vintage Books, New York, New York, 1980.

[Isr83] D. Israel, The Role of Logic in Knowledge Representation, *IEEE Computer 16*, 10 (1983), 37-42.

[KaK87] S. K. Karukonda and D. H. Kraft, A Survey on Natural Language Methods and Their Use in Information Retrieval Systems, *Tech. Report: #87-036*, Department of Computer Science, Louisiana State University, Baton Rouge, LA, 1987.

[KCP87] S. K. Karukonda, P. P. S. Chen and C. M. Parks, IMMS: An Expert System for Injection Mold Material Selection, *IEEE International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Hartford, Connecticut, October 19-20, 1987, 130-133.

[KaC87] S. K. Karukonda and P. P. S. Chen, Knowledge Programming in Prolog, *Tech. Report: #87-032*, Department of Computer Science, Louisiana State University, Baton Rouge, LA, 1987.

[KLC87] S. K. Karukonda, E. T. Lee and P. P. S. Chen, Design of a Pictorial Knowledgebase, *1987 ACM Computer Science Conference*, St. Louis, Missouri, February 17-19, 1987, 114-119.

[Kat50] D. Katz, *Gestalt Psychology*, The Ronald Press Company, New York, New York, 1950.

[KaF64] J. J. Katz and J. A. Fodor, The Structure of a Semantic Theory, *The Structure of Language (E. Bach and R.T. Harms Eds.)*, Englewood Cliffs, New Jersey, 1964.

[Ken78] W. Kent, *Data and Reality*, North Holland, Amsterdam, Netherlands,

1978.

[Ken79] W. Kent, Limitations of Record-Based Information Models, *ACM Transactions on Database Systems 4*, 1 (1979), 107-131.

[KKT76] L. Kerschberg, Klug and D. C. Tsichritzis, A Taxanomy of Data Models, *Systems for Large Data Bases (P.C. Lockman, and E.J. Neuhold, Eds.)*, Amsterdam, Netherlands, 1976.

[Ker87] L. Kerschberg, *Proceedings of the First International Conference on Expert Database Systems* , Benjamin Cummings, Menlo Park, California, 1987.

[KWD86] M. L. Kersten, W. Weigand, F. Dignum and J. Boom, A Conceptual Modelling Expert System, *5th International Conference on Entity-Relationship Approach*, Dijon, France, November 17-19, 1986, 275-288.

[Kes77] V. Kestenbaum, *The Phenomenological Sense of John Dewey*, Humanities Press, Atlantic Highlands, New Jersey, 1977.

[Kof63] K. Koffka, *Principles of Gestalt Psychology*, Harcourt, Brace & World, New York, New York, 1963.

[Koh47] W. Kohler, Gestalt Psychology, *An Introduction to New Concepts in Modern Psychology*, New York, New York, 1947.

[Kol82] G. Kolata, How Can Computers Get Common Sense, *Science 217* (September 24, 1982), 1237-1238.

[KoS85] H. F. Korth and A. Silberschatz, *Database System Concepts*, McGraw Hill, New York, New York, 1985.

[Kow74] R. Kowalski, Predicate Logic as a Programming Language, *Proceedings of IFIP Congress*, Stockholm, Sweeden, 1974.

[Kow79a] R. Kowalski, *Logic for Problem Solving*, North-Holland, New York, New York, 1979.

[Kow79b] R. Kowalski, Algorithm = Logic + Control, *Communication of the ACM 22* (1979), 424-436.

[KuP81] M. Kubovy and J. R. Pomerantz, *Perceptual Organization*, Lawrence Earlbaum Associates, Hillsdale, New Jersey, 1981.

[Laz87] R. Lazimy, Knowledge Representation and Modeling Support in Knowledge-Based Systems, *Proceedings of the Sixth International Conference on Entity-Relationship Approach*, New York, New York , November 9-11, 1987, 115-142.

[Lee85] E. T. Lee, Applications of the Entity-Relationship Approach to Similarity-Driven Pictorial Database Design, *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, October 28-10, 1985, 18-21.

[LKH88] E. T. Lee, S. K. Karukonda and S. Harajadi, Computer/Radar Joint Scheduling Using a Parallel Computer, *1988 ACM Computer Science Conference*, Atlanta, Georgia, February 23-25, 1988, 728.

[Lin86] P. Lindgren, Entities from a Systems Point of View, *5th International Conference on Entity-Relationship Approach*, Dijon, France, November 17-19, 1986, 3-15.

[Llo84] J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, New York, New York, 1984.

[LGH85] J. Ludewig, M. Glinz, H. Huser, G. Matheis, H. Matheis and M. F. Schmidt, SPADES - A Specification and Design System and its Graphical Interface, *Proceedings of the 8th International Conference on Software Engineering*, 1985, 83-89.

[Mar87] S. T. March, *Proceedings of the Sixth International Conference on Entity-Relationship Approach*, New York, New York, November 9-11, 1987.

[Mar85] A. P. Martinich, *The Philosophy of Language*, Oxford University Press, New York, New York, 1985.

[McH69] J. McCarthy and P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence 4 (B. Meltzer and D. Michie, Eds.)*, Endinburgh, UK, 1969, 463-502.

[McC77] J. McCarthy, Epistemological Problems in Artificial Intelligence, *Proceedings of IJCAI-77*, Cambridge, Massachusetts, 1977, 1038-1044.

[Mel85] C. S. Mellish, *Computer Interpretation of Natural Language Descriptions*, Ellis Horowood Limited, Chichester, England, 1985.

[Min68] M. Minsky, *Semantic Information Processing*, The MIT Press, Cambridge, Massachusetts, 1968.

[Min75] M. Minsky, A Frame Work for Knowledge Representation, *Computer Psychology (P.H. Winston Eds.)*, New York, New York, 1975.

[Min80] M. Minsky, K-Lines: A Theory of Memory, *Cognitive Science 4*, 2 (1980).

[Min85] M. Minsky, *The Society of Mind*, Simon and Schuster, New York, New York, 1985.

[MKR82] C. T. Morgan, R. A. King and N. M. Robinson, *Introduction to Psychology*, McGraw-Hill Publishing Co., New York, New York, 1982.

[New73] A. Newel, Production Systems: Models of Control Structures, *Visual Information Processing (W.G. Chase Eds.)*, New York, New York, 1973, 463-526.

[New82] A. Newel, The Knowledge Level, *Artificial Intelligence 18*, 1 (1982), 87-127.

[Ng80] P. A. Ng and A. F. Paul, A Formal Definition of Entity-Relationship Models, *Entity-Relationship Approach to Systems Analysis and Modeling (P.P.S. Chen, Eds.)*, Amsterdam, Netherlands, 1980, 103-119.

[Ng81] P. Ng, Further Analysis of the Entity-Relationship Approach to Database Design, *IEEE Transactions on Software Engineering SE-7*, 1 (1981), 85-95.

[Nil81] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, California, 1981.

[Per84] F. Pereira, C-Prolog User's Manual, Version 1.5, EdCAAD , Department of Architecture, University of Edinburgh, Edinburgh, UK, 1984.

[Pol69] M. Polanyi, *Knowing and Being*, The University of Chicago Press, Chicago, Illinois, 1969.

[Rei78] R. Reiter, Deductive Question-Answering on Relational Data Bases, *Logic and Data Bases (H. Gallaire and J. Minker Eds.)*, New York, New York, 1978.

[Ren82] T. Rentsch, Object-Oriented Programming, *SIGPLAN Notices Notices 17*, 9 (1982), 51.

[RoG77] R. B. Roberts and I. P. Goldstein, The FRL Primer, *Report AIM-408*, AI Lab, Massachusetts Institute of Technology, Cambridge, MA, 1977.

[Rob65] J. A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle, *Journal of the ACM 12*, 1 (January 1965), 23-41 .

[RoM75] N. Roussopolos and J. Mylopoulos, Using Semantic Networks for Database Management, *Proceedings of the International Conference on Very Large Data Bases*, 1975, 144-172.

[Sak80] H. Sakai, Entity-Relationship Approach to the Conceptual Schema Design, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1980, 1-8.

[SNF80] C. S. D. Santos, E. J. Neuhold and A. L. Furtado, A Data Type Approach to the Entity-Relationship Model, *Entity-Relationship Approach to Systems Analysis and Modeling (P.P.S. Chen, Eds.)*, Amsterdam, Netherlands, 1980, 103-119.

[Sch72] R. C. Schank, Conceptual Dependency: A Theory of Natural Language

Understanding, *Cognitive Psychology 3* (1972), 552-631.

[ScA77] R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals, and Understanding*, Lawrence Earlbaum, Hillsdale, New Jersey, 1977.

[ScS79] G. Schiffner and P. Scheuermann, Multiple View and Abstractions with an Extended Entity-Relationship Model, *Journal of Computer Languages 4* (1979), 139-154.

[SeS85] A. Sernadas and C. Sernadas, The Use of E-R Abstractions for Knowledge Representation, *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, October 28-30, 1985, 224-231.

[Ske87] R. R. Skemp, *Psychology of Learning Mathematics*, Lawrence Earlbaum Associates, Hillsdale, New Jersey, 1987.

[SmS77] J. M. Smith and D. C. P. Smith, Database Abstractions: Aggregation and Generalization, *ACM Transactions on Database Systems 2*, 2 (1977), 105-133.

[Sow84] J. F. Sowa, *Conceptual Structures: Information Processing in Minds and Machines*, Addison-Wesley, Reading, Massachusetts, 1984.

[StC85] L. Steels and J. A. Campbell, *Progress in Artificial Intelligence*, Ellis Horowood Publishers, Chichester, England, 1985.

[Ste79] M. Stefik, An Examination of a Frame-Structured Representation System, *Proceedings of IJCAI-79*, Tokyo, Japan, 1979, 845-852.

[StS86] L. Sterling and E. Shapiro, *The Art of Prolog*, The MIT Press, Cambridge, Massachusetts, 1986.

[TeF82] T. J. Teory and J. D. Fry, *Design of Database Structures*, Prentice Hall , Englewood Cliffs, New Jersey, 1982.

[Tou86] D. S. Touretzky, *The Mathematics of Inheritance Systems*, Pittman, London, England, 1986.

[TsL82] D. C. Tsichritzis and F. H. Lochovsky, *Data Models*, Prentice Hall, New York, New York, 1982.

[Ull82] J. D. Ullman, *Principle of Database Systems*, Compute Science Press, Rockville, Maryland, 1982.

[Wal84] M. Wallace, *Communicating with Databases in Natural Language*, Ellis Horowood Limited, Chichester, England, 1984.

[Wat86] D. A. Waterman, *A Guide to Expert Systems*, Addison Wesley, Reading, Massachusetts, 1986.

[Wer71] M. Wertheimer, *Productive Thinking*, Harper & Row Publishers, New York, New York, 1971.

[Wie84] G. Wiederhold, Knowledge and Database Management, *IEEE Software*, January 1984, 63-73.

[Win83] T. Winograd, *Language as Cognitive Process, Vol. 1*, Addison-Wesley, Reading, Massachusetts, 1983.

[WiH84] P. H. Winston and B. K. P. Horn, *Lisp*, Addison Wesley, Reading, Massachusetts, 1984.

[Win84] P. H. Winston, *Artificial Intelligence*, Addison Wesley, Reading, Massachusetts, 1984.

[Woo75] W. A. Woods, What is a link: Foundations for Semantic Nets, *Representation and Understanding (D.G. Bobrow and A. Collins Eds.)*, New York, New York, 1975, 35-82.

[Woo83] W. A. Woods, What's Important about Knowledge Representation, *IEEE Computer 16*, 10 (1983), 22-29.

[WRC65] L. Wos, G. Robinson and D. F. Carson, Automatic Generation of Proofs in the Language of Mathematics, *Proceedings of IFIP Congress 65 2* (1965), 325-326, Barton Books.

[WOL84] L. Wos, R. Overbeek, E. Lusk and J. Boyle, *Automated Reasoning: Introduction and Application*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[Zus70] L. Zusne, *Visual Perception of Form*, Academic Press, New York, New York, 1970.

Vitae

Sreerama K. Karukonda has been a graduate student at Louisiana State University since August 1984. His major area of research is integrating artificial intelligence and data base techniques. His other research interests include neural nets, programming languages, and software engineering.

He earned Master of Science degree in Systems Science from LSU in 1986. He obtained Master of Technology degree in Mechanical Engineering from Indian Institute of Technology, Khargpur, India in 1981. He had his Bachelor of Engineering degree from Andhra University, Waltair, India in 1979. He worked as a research engineer in Enginnering Research Center, TELCO, India between August 1981 and August 1982.

He is married to Vimala Devi. After graduation, he is taking up an assistant professor position in the Department of Computer Science at Northeast Louisiana University.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Sreerama K. Karukonda

Major Field: Computer Science

Title of Dissertation: Entity-Relationship Approach To Knowledge Base Systems

Approved:

Peter Chen
Major Professor and Chairman

F. Glen Kemery
Dean of the Graduate School

EXAMINING COMMITTEE:

Doris L. Carver

David A. Kay

Edward T. Lee

Charles M. Parks

Date of Examination:

July 6, 1988