AN EXTENSION OF A FIRST-ORDER LANGUAGE AND ITS APPLICATIONS

by
Dong-Guk Shin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1985

Doctoral Committee:

Professor Keki B. Irani, Chairman
Professor Andreas R. Blass
Professor Yuri Gurevich
Professor Arch W. Naylor
Associate Professor Toby J. Teorey

For my mother and father,
my sisters and brother

ACKNOWLEDGMENTS

The author owes lasting debts to many people in completing this endeavor.

My deepest gratitude goes to Prof. Keki Irani for his untiring encouragement, faithful guidance and occasional role as devil's advocate; to Prof. Andreas Blass for invaluable discussions and unforgettable aid; to Professors Toby Teorey, Yuri Gurevich and Arch Naylor for their time and acute critiques; and to Professors Norman Scott and William Root for their warm support and encouragement at the early stage of my graduate study.

My thanks are extended to my colleagues and friends, Chin-Wan Chung, Mourad Oulid-Aissa, Genesio Hubscher, Carol Luckhardt, Yi-fong Shih, Suk In Yoo, and Pamela Walters, who made this endeavor more than bearable.

My thanks also go to Elizabeth Olsen and Virginia Folsom who typed a large section of Part I; to Prof. Thomas Sawyer, Betty Cummings, and my friend Gretchen Antelman who helped me make the manuscript more readable.

My gratitude is also extended to the people who made available the financial resources for this endeavor which include fellowships of the CICE graduate program and grants from AFOSR under contract F49620-82-C-0089.

Finally, my thanks go to my parents, Byung-Sik Shin and Chun-Soo Kim, and my sisters and brother. Without their endless support, I could neither have started nor have completed this work. This work is dedicated to them.

D. G. S.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF APPENDICES

Appendix

CHAPTER I

INTRODUCTION

1.1. Motivation

When deductions are made in certain axiomatic systems involving more than one category of objects (e.g., points, lines and planes), two approachs are available: (i) a many-sorted logic in which there are distinct kinds of variables for the different categories of objects, and (ii) a one-sorted logic in which there is only one kind of variable for all categories of objects, but in which there are special predicates to effect the range restriction of the variables to the respective categories of objects. These two approachs are equivalent in the sense that deduction made by one approach can also be made by the other approach†.

In spite of their equivalence, many-sorted logic offers various advantages over one-sorted logic. For example, many-sorted logic allows the utilization of sortal information to enhance the deduction efficiency, and the language for many-sorted logic allows a more compact expressive power than does the language for a one-sorted logic. These advantageous features were originally observed by Herbrand who

† Their equivalence is formally shown by the Herbrand-Schmidt theorem [Herb30, Schm38]. Let T_n $(n = 2, \cdots, \omega)$ be a many-sorted system, and let $T_1^{(n)}$ be its corresponding one-sorted system. In [Wang52] Wang mentions "In [5] Herbrand states a theorem which amounts to the following (see [5], p.64): (I) A statement of any system T_n is provable in T_n if and only if its translation in the corresponding system $T_1^{(n)}$ is provable in $T_1^{(n)}$. However, the proof he gives there is inadequate, failing to take into account that there are certain reasonings which can be carried out in $L_1^{(n)}$ but not in L_n. In [1], Arnold Schmidt points this out and devotes his paper to giving a careful proof of the theorem."

first proposed many-sorted logic in his thesis [Herb30]. Following him, various versions of many-sorted logic were proposed and investigated by Schmidt [Schm38, Schm51], Wang [Wang52], Hailperin [Hail57], and Idelson [Idel64].

Recently, the advantages of many-sorted logic have been explored in various areas of computer science including the fields of database design and automatic theorem proving. In the database design area, many-sorted logic is used as a means of formalizing the database [McMi77, GaMi78, Reite81], and in the automatic theorem proving area, many-sorted logic is used to increase deductive efficiency [Weyh77, Cham78, Cohn83, Walt83, Walt84a, Walt84b].

Although many-sorted logic appeals to various applications of computer science because of the advantages it offers, usage of many-sorted logic is often restricted to a certain extent. The following situation is considered: a system involving more than one category of objects is axiomatized based on many-sorted logic, and the categories of objects determine the sort structure of the axiomatized system. After the sort structure is determined and when deduction is made in the system, it turns out that a new sort is needed that does not exist in the previously determined sort structure. At this moment, the sort structure determined beforehand can be changed to accommodate the new sort, but in some situations it may not be desired to do so for various reasons. When the sort structure determined a priori is desired not to be changed, a variable ranging over a new sort cannot be introduced in the currently known many-sorted logic.

In this thesis an extended predicate calculus is proposed in which the problem described previously is avoided. The extended predicate calculus is obtained by embedding a new kind of syntactic object called *an aggregate variable* in the first-order language. Then in this extended predicate calculus, variables whose interpretations are restricted by arbitrary ranges can be introduced as freely as needed during deduction without changing the sort structure determined a priori.

Informally speaking, the aggregate variables are syntactically ordinary sort variables, but semantically they are variables whose ranges are restricted by unary relations instead of sorts. Therefore, whenever aggregate variables are introduced, the sort structure does not need to be changed; the system only needs to be augmented by new unary relations that will be the respective ranges of interpretation of the aggregate variables. This property of the extended predicate calculus is called *Σ-extensibility.*

When aggregate variables are introduced as part of the first-order language, they can be embedded in a one-sorted language as well as in a many-sorted language. In the former case, the resulting language is called *a one-sorted language with aggregate variables*, denoted by L^1_Σ, and in the latter case, the resulting language is called *a many-sorted language with aggregate variables*, denoted by L_Σ.

1.2. Objectives

The objectives of the thesis are twofold: (i) to provide the theoretic foundation for the extended predicate calculi, and (ii) to demonstrate their practical usage in real applications.

Concern for the first part is with the syntax of each of the two languages L_Σ and L^1_Σ, their interpretations and their Σ-extensibilities. For the second part, two applications have been chosen that demonstrate the practical usage of L_Σ and L^1_Σ, respectively. One of these applications is related to the distributed database design area and the other, to the automatic theorem proving area.

In the first application, L_Σ is used as a tool to describe the user queries to the database and the knowledge about the database. Here it is demonstrated that L_Σ offers a more compact expressive power than an ordinary many-sorted language, which therefore allows the development of a methodology to partition relations horizontally in the context of the distributed database design. In the second application, L_Σ^1 is used as a tool to describe a many-sorted theory. In this case, it is shown that L_Σ^1 allows the introduction of variables whose ranges are restricted to new sorts in the middle of refuting the many-sorted theory, which implies a more efficient many-sorted resolution scheme than the currently existing one.

The rest of the thesis naturally divides into two parts: one for the application of L_Σ and the other for the application of L_Σ^1. Part I deals with the application of L_Σ in the distributed database design area and Part II, the application of L_Σ^1 in the automatic theorem proving area. Part I consists of Chapters I through VII and Part II consists of Chapters VIII through XII. Conclusions of the thesis are given in Chapter XIII.

PART I

In this part, a knowledge-based approach is proposed with which the user reference clusters of a database are estimated which can be used in partitioning a relational database horizontally during distributed database design. Using the knowledge about the data, the user queries are converted to equivalent queries by a proposed inference procedure. The user reference clusters estimated from these revised queries are more precise than those that can be estimated from the original user queries. A many-sorted language with aggregate variables (L_Σ) is used for the representation of the user queries and the knowledge base. The types of knowledge to be used are discussed. An example illustrates the way inference is carried out, and the correctness of the inference is also discussed.

CHAPTER II

PARTITIONING A RELATIONAL DATABASE HORIZONTALLY USING A KNOWLEDGE-BASED APPROACH

2.1. Introduction

Since the notion of a distributed system (DS), as distinct from a centralized system, was introduced, computer scientists have focused a great deal of attention on the well-defined problems of a distributed system, such as file allocation and network design [Chu69, Whit70, Case72, Chu73, MaRi76, IrKh79]. With the advent of distributed database systems (DDBS), especially when the data model is relational, data allocation in DDBS has been interpreted in a different way from that of file allocation in a DS [RoGo77].

In the file allocation problem the main issue is how to transfer the characteristics of a distributed system into the parameters of a cost optimization model so that the optimum allocation of files could be determined from the model. This view was based on the assumption that files, or relations, are independent of each other; in other words, only one file is needed to answer each query issued at each site. That means that whenever the queried file does not reside at the query site, to answer the query, either the file is transferred to the query site or the query is sent to the file site and the answer is sent back to the query site.

In the data allocation problem, however, files, or relations, are no longer regarded as independent. Due to the logical intricacy among the relations, processing a query involves one or more relations which requires costly intermediate network processing if all the relations queried are not locally available. Consequently, to include the network flow caused by the intermediate processing in a DDBS design model, the logical relationships among data should be somehow reflected in the model of data allocation. For this reason, the issue of data allocation during the design of a DDBS is different from that of file allocation.

The current trend in the design of a DDBS is to partition the relations horizontally and/or vertically and to allocate the fragments of relations over a network [WoKa83, CeNW83, Ouli84]. In these studies, therefore, each local database of a DDBS consists of horizontally partitioned or vertically partitioned fragments of relations instead of complete copies of the relations.

The benefits of assigning the fragments of relations have been well understood [RoGo77, TeFr82]. Partitioned fragments offer a great deal of flexibility in distributing data so that the **user reference clusters** (*URC*'s) to the database at each site -- which means certain portions of the relations, or files, of the database around which queries are clustered -- could be faithfully reflected in the distribution of data. Thus, with the appropriate replication of fragments, total network flow is reduced and the probability of parallelism in distributed query processing is increased, while the update cost induced by replication is confined to the replicated fragments.

In spite of realizing that such benefits accrue from allocating partitioned fragments, not much work has been done in this area, especially in the area of partitioning relations horizontally and distributing their fragments. A major difficulty here is that there is no known significant criteria that can be used to partition relations horizontally. An often suggested practice, for example [WoKa83], is to analyze the expressions for the user queries at each site. These expressions may reveal the *URC*'s to the database and thus these *URC*'s can be used as a means to partition the relations horizontally. However, there is a problem even in this approach because the information contained in the user queries is not sufficient to estimate the *URC*'s precisely. When the *URC*'s are not identified accurately, they may result in an inadequate partitioning. For this reason, determining the *URC*'s as precisely as possible is a well-defined issue in the horizontal partitioning problem.

In Part I, an approach is suggested for better estimating *URC*'s by utilizing not only the user query expressions but also certain knowledge about the data itself. The intended approach is illustrated in the following example.

Example 2.1.1

A database of a big auto corporation is used in this example. Let *DIVISIONS* , *DEALERS* , and *SALES* be the relations where *DIVISIONS* keeps the information about all the divisions of a big auto corporation such as assembly plants, parts plants, and headquarters; *DEALERS*, the information about all the dealers with which the corporation has transactions; and *SALES*, all the sales transactions between the plants and the dealers.

Suppose there is a query originating frequently at car assembly plants that asks for information about the purchasers of car items, for instance, "What are the addresses of the dealers who were supplied item# B47, V01, or V03? " where the item#'s $B47$, $V01$, and $V03$ stand for some car items. Based on this query, a

DB designer may try to identify the *URC*'s to the relations *SALES* and *DEALERS* and eventually utilize the *URC*'s in partitioning *SALES* and *DEALERS*. However, the *URC*'s cannot be determined precisely enough solely from the query. That is, although it is determinable that at car assembly plants the references to the relation *SALES* are clustered on the fraction of some transactions of car items, say $SALES[B47, V01, V03]$, no cluster of references can be assessed on *DEALERS* because no restrictions have been imposed on the dealers in the query.

Suppose there is a fact about this database expressed in English as, "All car purchasers should be car dealers," which implies a relationship between some tuples of *SALES* and some tuples of *DEALERS*, or simply between a fraction of *SALES*, namely, *CAR_SALES*, and a fraction of *DEALERS*, namely, *CAR_DEALERS*. It can then be postulated that such knowledge can be utilized for estimating better *URC*'s than the previous one which was obtained solely from the query. That is, by knowing that only car dealers purchase car items, it can be concluded that only the fraction of *CAR_DEALERS* would be queried at the car assembly plants.

The preceding example shows that a DB designer can utilize some knowledge about the data in an effort to identify the *URC*'s as precisely as possible. In Part I, it is intended to formalize the DB designer's role by constructing what is called a knowledge-based system (KBS). The function of the KBS will then be to determine the *URC*'s from the user provided query expressions by applying the knowledge about the database.

Once the *URC*'s are identified, determining horizontal partitions of relations from these estimated *URC*'s can be done straightforwardly. That is, each relation can be partitioned in terms of the *URC*'s identified for that relation. For instance, in the preceding example, *DEALERS* can be partitioned into *CAR_DEALERS* and *DEALERS−CAR_DEALERS*, and *SALES*, into *SALES*[*B*47,*V*01,*V*03] and *SALES−SALES*[*B*47,*V*01,*V*03]. This is a legitimate way to partition relations in the sense that as far as processing the query of the example is concerned, the other fractions of the relations are irrelevant. Once the partioning is completed, the fragments can be treated as separate objects for an optimal allocation which would assure the benefits of a horizontally partitioned distributed database design.

The overall DDBS design scheme can be viewed as a conjunction of two separate subcomponents, namely, a horizontal partitioning system and a mathematical programming model. The former is a front-end system based on a knowledge-based approach that produces the unit objects to be dispersed, i.e., the horizontally partitioned fragments of relations, and the latter, a linear or nonlinear programming model that determines the optimal distribution of the unit objects. Because the knowledge-based approach is employed to determine the unit objects of distribution, this DDBS design scheme is called a knowledge-based distributed data base design (KBDDBS design). The schematic diagram for the KBDDBS design is shown in Figure 2.1.

As a quantitative cost optimization model, the second subcomponent must be furnished with two key input parameters. They are:

(1) The unit objects to be dispersed over a network.

(2) The frequency with which each unit object is queried at each site.

Then, given a set of queries, the total network flow for each allocation configuration can be estimated with some distributed query processing algorithm as discussed in

[Bern81, Chun83] and, therefore, the optimal configuration of the horizontally partitioned fragments can be determined. As far as developing a mathematical programming model is concerned, however, there has been much work in the context of file allocation [MaRi76, MoLe77, FiHo80, Ouli84, DoFo82]; therefore, some adaptation of any of these studies would suffice. For this reason, the mathematical programming model part will not be taken into consideration in this work.

Employing a knowledge-based approach that constitutes the first subcomponent is, therefore, the major concern in this work. The goal of employing the knowledge-

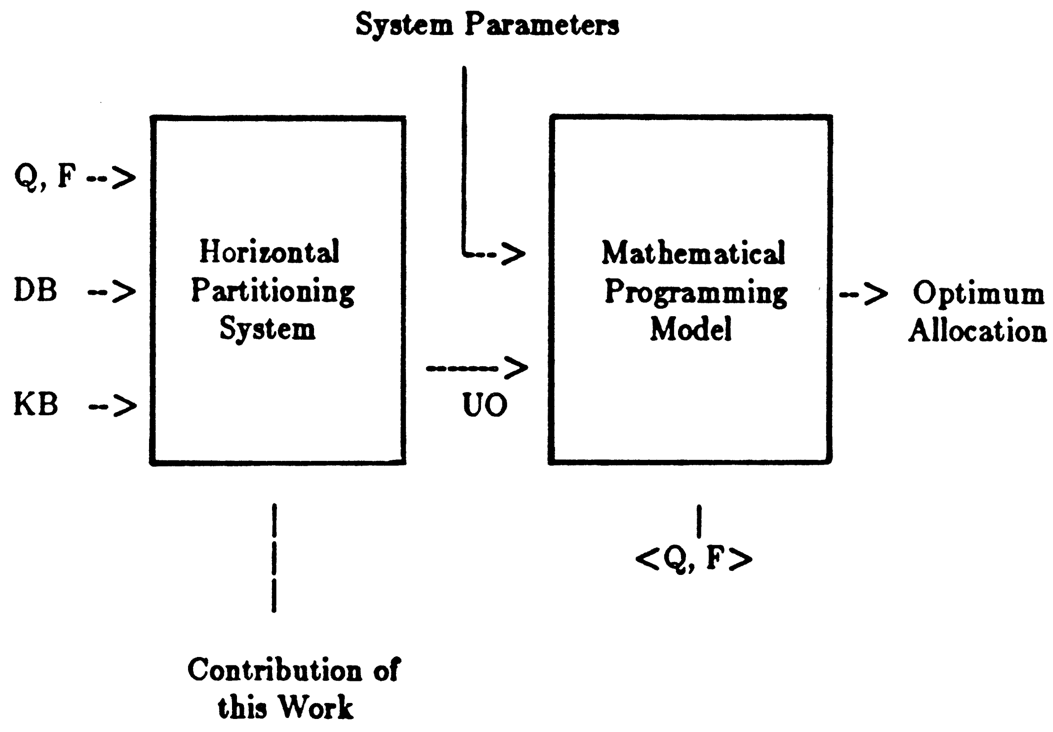

Figure 2.1. Framework of the KBDDBS Design

based system as the front end of a DDBS design is, as stated previously, to exploit the knowledge of the data for partitioning relations horizontally to best suit distribution over a network. In realizing such a goal, there are three issues to be addressed:

(1) How the user queries and the knowledge should be expressed so that the knowledge can be applied to the user queries in a deductive way.

(2) What types of knowledge should be utilized for this purpose.

(3) How the inference should be carried out.

The rest of this part deals with these three issues.

2.2. Related Literature

In this section, the current research which is related to our study is briefly reviewed. The related research is discussed in three contexts: what techniques of horizontal partitioning of relations have been developed in designing a database?; how has the notion of horizontal partitioning of relations been employed in designing a DDBS ?; and finally, what are the current techniques of AI and how have the techniques of AI been used in a database design?

Partitioning a relation vertically and/or horizontally is well understood [Ullm80, TeFr82, Date83]. Much has been made of the vertical partitioning of relations to achieve efficient and secure data manipulation, for example, removing redundancy and update anomalies from a database. In the context of designing a database, however, less attention has been paid to horizontal partitioning. Most recently, although their applications are limited to some extent, there have been several attempts, to develop a theory of horizontal partitioning analogous to the well conceived normalization theory, such as [Bern76, Delo78], so that more secure data manipulation can be assured than when only vertical partitioning is applied.

In [Furt81], a technique has been developed so that a relation, some of whose key attributes are determined by a non-key set of attributes, may be converted into Boyce-Codd normal form by partitioning relations horizontally prior to the conversion which is otherwise impossible. In [DePa82], it has been shown that for some classes of relations, a larger class of functional dependencies could be revealed by starting with horizontal partitioning and, therefore, with the additionally detected functional dependencies, more powerful vertical partitioning of the relations may be accomplished.

In the context of designing a DDBS, the idea of partitioning relations horizontally as well as vertically, has been initiated in the early distributed database design work [RoGo77]. In [EpSW78], a query processing algorithm which exploits a parallelism in a distributed environment has been discussed. In their algorithm, the parallelism in a query processing is sought by partitioning relations horizontally and replicating the fragments over several sites except the relation whose partitioning and replication promises the least storage cost efficiency. Most recently in conjunction with maintaining a DDBS which composed of horizontally partitioned physical fragments in a distributed environment, [MaUl83] has suggested some algorithms for inserting and deleting tuples from the fragments.

As the first significant work in designing a horizontally partitioned DDBS, a design methodology for a distributed database in which each local database is not a collection of relations but a collection of the fragments of relations has been initiated in [Wong81, WoKa83]. In their work, the semantics of the logical schema reflected in a class of queries are exploited as a means of partitioning relations and from this partitioning, data are distributed in a specific way, which is called "locally sufficient," in order to suppress network flow by employing a high degree of parallelism in processing queries. Their method, however, has various shortcomings, especially when a real environment is not faithfully reflected in their model: first, maintaining that local sufficiency involves prohibitive levels of update cost unless the database is strictly static; second, the communication cost of collecting the final results at the site where the query originated would cost more than the benefits gained from parallel processing, unless the communication cost is far less than the processing cost; and third, while each site's response time may be shortened, the total system throughput may be decreased unless system job loads among the nodes of network are evenly distributed and managed all the time. In short, though it depends on the characteristics of a database and the system parameters, the parallelism in a query processing over a remotely dispersed computer network may not achieve the benefits which are usually obtained in the parallel processing with a tightly coupled multiprocessors, mainly because of the high costs of network communication and the maintenance of local sufficiency at all times.

In contrast to the above studies, the design objective of this work is not confined to parallelism in a distributed query processing. Rather our data allocation scheme is based on the philosophy that the minimization of total network communication cost should be achieved by appropriate replication of horizontally partitioned fragments of relations instead of complete copies of relations. By doing so, the *URC*'s at each site are faithfully reflected; and, therefore, the user queries may be processed as locally as possible; and, furthermore, the parallelism in query processing can also be achieved because of the high degree of replication. The price paid in this approach is the storage and update cost of the replicated fragments. However, it is expected that since the update cost of replicated data shrinks as much as the size of the replicated parts of relations shrinks, there is much more leeway to replicate fragments than when fragmentation is not considered.

The main issue in our approach is how to take advantage of the knowledge about the data in partitioning relations. As has been pointed out in the previous section, our approach resorts to AI techniques, i.e., drawing inferences from the knowledge about the data and the user queries. In the following, it is first briefly reviewed what techniques have been developed in AI and then it is discussed how the AI techniques have been used in the context of designing a database.

With the assumption that all the knowledge to be used is known -- aside from the problem of knowledge acquisition -- the problem of AI is in general divided into two parts. One is how to represent the knowledge and the other is how to utilize the knowledge once it has been represented†. The classical approach to representing the knowledge has been *formal logic*. The modification of formal logic from a working tool for philosophers' and mathematicians' into a knowledge representation tool in AI has been initiated by the development of automatic theorem proving techniques, such as the resolution principle [Robi65a]‡, Here formal logic is used as a knowledge representation formalism and the resolution principle is used as an inference mechanism. The important features of logic are the preciseness† in expressing the

---

† In [McHa69], the problem in AI is differentiated into an epistemological part and a heuristic part. In his classification, the problem of knowledge acquisition is included in the epistemological part.

‡ An algorithm to find an interpretation that can falsify a given formula has been invented by Herbrand in 1930. Gilmore, in 1960, first tried to implement Herbrand's procedure on a computer which turned out to be very inefficient [Gilm60]. Few months later, Davis and Putnam published improved version of Gilmore's program which still was not efficient enough [DaPu60]. A major breakthrough was made by Robinson's resolution principle in 1965 which was much more efficient than any earlier procedure [Robi65a].

knowledge and the correctness in inferring any conclusion. Various AI systems based on logic have been suggested, including a general-purpose question-answering system QA3 [Gree69], a robot planning system STRIPS [FiHN72], and a proof checker for proofs stated in first-order logic FOL [FiWe76]. The current research in logic includes the development of a more efficient inference mechanism such as theorem proving via general mating [Andr81], and an extension of the first order logic, such as fuzzy logic [Lee72] -- in which how common sense and intuition can be handled are major concerns.

The major consideration of logic as a representational tool was how the knowledge identified as useful in the problem domain could be adequately and precisely represented. Departing from this view, a new interpretation about the knowledge has been initiated by a group of researchers, called proceduralists, who argue that the way to use the knowledge -- how to make inferences -- should also be explicitly included in the knowledge to be represented. A representation scheme, called *procedural representation,* has been suggested and its emphasis was on how to express the procedural knowledge -- the control information for inferences -- in a better way. The advantage of this representation scheme is that the inefficiency in processing knowledge represented in logic could be avoided. Starting with PLANNER [Hewi72], a number of procedural representation-based AI programming language projects have followed, including CONNIVER [SuMc72], QA4 [RuDW72], POPLER [Davi72], and QLISP [Rebo76].

Another descriptive purpose-oriented knowledge representational formalism, called *semantic networks,* has been initiated [Quil68, NoRu75, AnBo73]. The

† In [Haye77], a complete discussion of this issue is presented and the advantage of logic over other representation systems on these grounds is argued.

problem in a semantic network is that no simple set of unifying principles is available due to its diversified development. Semantic networks, however, became very popular in AI because of the graphical representation which resembles human memory association. The first program to use semantic network techniques in AI was a question-answering system SIR [Raph68] which was followed by SCHOLAR [Carb70]. Several semantic network "languages" have been proposed which have the full expressive power of predicate calculus. The examples are network formalism [Schu76], partitioned semantic network formalism [Hend75], and the SNePS system [Shap79].

Because of its modular knowledge representation facility -- describing the knowledge about what to do in a specific situation -- what is increasing in popularity is *production system* which has been developed by Newell and Simon [NeSi72]. The basic idea of these systems is that a knowledge base consists of rules, called productions, in the form of condition-action pairs: "If this condition occurs, then do this action." The major problem of this representation formalism is the inefficiency of program execution. The strong modularity and uniformity of the productions results in high overhead when they are used in problem solving. Despite the inefficiency of programming execution, because of its naturalness -- statements about what to do in predetermined situations are naturally encoded into production rules -- production systems have been used as the backbone of expert systems like DENDRAL [BuFe78], MYCIN [Shor76], and PROSPECT [DuHN76].

Most recently a knowledge representation scheme, called *frame,* which facilitates "expectation-driven processing" has been proposed by Minsky [Mins75]. The important feature in frame is that the procedural knowledge can be easily incorporated into the representations in this scheme: procedures can be attached to slots to incorporate the reasoning or problem-solving behavior of the system. The current AI systems based on frame include KRL [Bowi77], NUDGE [GoRo77], and KLONE [Brac78]. Many researchers in AI, however, have different ideas about what a frame is: there are many fundamental differences in approach among the researchers who have designed frame-based systems.

As the two research areas DB and AI grow, the researchers of both areas begin to recognize the common realm shared by the two areas and start to exchange ideas and techniques [SAMP81]. In the following, it is briefly reviewed what AI techniques have been employed in the context of database system design.

Historically Query-Answering(Q-A) system [Chan76, Mink78, Reit78b] has long relied on the automatic theorem proving(ATP) technique where the query to be answered is submitted as a conjectured theorem to be proved. With the advent of database management systems(DBMS), several works, such as LADDER [Hend78], PLANES [Walt78], and RENDEZVOUS [Codd78], applied AI techniques to the natural language interfaces as a part of DBMS. Most recently, [HaZd80] and [King81] suggested semantic query optimization processing which departs from the conventional approach of [WoYo76, Yao79]. What is noticeable in their work is that the knowledge about the data and the information about the file structure are explicitly used to transform the original query into an equivalent new one which promises far more efficient processing.

One important aspect to be considered in these systems is the role of the knowledge-based system which is being employed in each system. In Q-A system, the whole system itself is viewed as a knowledge-based system, which means all the facts of a conceived real world are represented in some knowledge representation language to form a database and the resolution principle is employed as its inference mechanism. In natural language query systems, however, the knowledge-based system is just a front-end mediator to support the translation of a natural language query into a formal form of query. Compared to these, the knowledge-based system of semantic query optimization is regarded as an expert system which guides the transformation of the original query into a new one whose processing cost is far less than the processing cost of the original one.

As distinct from any of these systems, the knowledge-based approach in our study is to assist the design of a DDBS.

2.3. Organization

The rest of Part I is organized in the following way:

In Chapter III, the theoretical foundation for embedding the aggregate variables in an ordinary many-sorted language is established. Here a many-sorted language with aggregate variable (L_Σ) is introduced in three contexts: syntax of L_Σ, interpretation of L_Σ, and the Σ-extensibility of L_Σ.

In Chapter IV, by using the extended calculus, a database and a knowledge base associated with that database is modeled as a logical structure and a theory, respectively. Then it is shown how the knowledge-based approach is employed by constructing a knowledge-based system (KBS). The issues about constructing the KBS are discussed in detail.

In Chapter V, the notion of scheduled user queries is introduced as a design resource. Then L_Σ is used as a tool to represent the scheduled user queries, i.e., a specific form of L_Σ called Σ-normal form is suggested as the representation formalism for the scheduled user queries.

In Chapter VI, the types of knowledge to be used in the knowledge-based approach are identified in terms of five axiom schemas in L_Σ. Instances of these schemas constitute the knowledge base, denoted KB, of the KBS. Also, the notion of a Σ-Horn knowledge base, denoted $KB_{\Sigma H}$, is introduced as a specific class of knowledge of KB for later use.

Finally, in Chapter VII an inference procedure is suggested as a tool to apply the elements in the $KB_{\Sigma H}$ to the scheduled user queries of the Σ-normal form to derive the URC's. The soundness of the inference is discussed. How the relations can be partitioned based on the estimated URC's is also discussed briefly. Conclusions and the direction of future work are also given.

In Appendix A, a fraction of a relational database example is shown that is used as the master example throughout this part.

CHAPTER III

MANY-SORTED LANGUAGE WITH AGGREGATE VARIABLES L_Σ

3.1. Syntax of L_Σ

In this chapter, a language of an extended predicate calculus, called a many-sorted language with aggregate variables (L_Σ), is introduced. L_Σ is a formal language obtained by embedding a new syntactic object, called aggregate variable, in an ordinary many-sorted language (L_m). In this section the syntax of L_Σ is first introduced.

In L_Σ, two types of variables, called simple variables and aggregate variables, are available. The simple variables of L_Σ are the same as the sort variables of L_m. The aggregate variables are syntactically ordinary sort variables, but semantically they are variables whose ranges are restricted to unary relations instead of sort domains. Let L_Σ be with a sort index set I. Formally stated, an *aggregate variable of sort* $i \in I$ is of the form $x_i{}^{\Sigma Q}$ where Q is a unary predicate symbol of sort i in L_Σ. Semantically $x_i{}^{\Sigma Q}$ ranges over the unary relation indicated by Q which is a subset of the domain of sort i.

L_Σ with a sort index set I is formally defined in the following:

Definition 3.1.1

A many-sorted language with aggregate variables L_Σ consists of the following: (1) $|I|$ infinite disjoint sets $V^1, \cdots, V^{|I|}$ where the elements of V^i, $1 \leq i \leq |I|$, are called simple variables of sort i ; (2) $|I|$ infinite disjoint sets $V_a^1, \cdots, V_a^{|I|}$ where the elements of V_a^i, $1 \leq i \leq |I|$, are called aggregate variables of sort i ; (3) $|I|$ disjoint sets $C^1, \cdots, C^{|I|}$ where the elements of C^i, $1 \leq i \leq |I|$, are called constant symbols of sort i ; (4) for each n-tuple $<i_1, \cdots, i_n>$, $\{i_1, \cdots, i_n\} \subseteq I$, a set $R^{<i_1, \cdots, i_n>}$ whose elements are called predicate symbols of sort $<i_1, \cdots, i_n>$; (5) for each $n+1$-tuple $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, a set $F^{<i_1, \cdots, i_n, i_{n+1}>}$ whose elements are called function symbols of sort $<i_1, \cdots, i_n, i_{n+1}>$; (6) logical connectives $\neg$ and $\rightarrow$; and (7) a universal quantifier $\forall$. ●

When it is convenient, L_Σ will be represented as a quintuple, $L_\Sigma = <P, R, F, C, \rho>$ where P is a unary predicate set whose members are exclusively used in the superscripts of aggregate variables, R is a predicate symbol set, F is a function symbol set, C is a constant symbol set and ρ is the arity function such that $\rho : R \cup F \rightarrow N^+$, where N^+ is the set of positive integers. In the tuple-representation of L_Σ, P and R may not necessarily be disjoint. If a unary predicate symbol, say Q, in R is used in the superscript of an aggregate variable, Q is also a unary predicate symbol belonging to P.

The syntax rule of L_Σ is given in the following. First, the set of *terms of sort* i is inductively defined as follows: (i) any simple or aggregate variable of sort i or constant symbol of sort i is a term of sort i, and (ii) if f is a function symbol of sort $<i_1, \cdots, i_n, i_{n+1}>$ and $t_1, \cdots, t_n$ are terms of sort $i_1, \cdots, i_n$, respectively, then $f(t_1, \cdots, t_n)$ is a term of sort i_{n+1}. An *atomic formula* of L_Σ is defined to be a sequence of the form $A(t_1, \cdots, t_n)$ where A is an n-place predicate symbol of sort $<i_1, \cdots, i_n>$ and t_j, $1 \leq j \leq n$, is a term of sort i_j. Let the set of *atomic formulas* of L_Σ be denoted by $Atom(L_\Sigma)$. The set of *well-formed formulas* of L_Σ, $Form(L_\Sigma)$, is then defined recursively as: (i) if $\alpha \in Atom(L_\Sigma)$, then $\alpha \in Form(L_\Sigma)$; (ii) if $\alpha, \beta \in Form(L_\Sigma)$, then so are $\neg \alpha$, $(\alpha \rightarrow \beta)$, and $\forall v\ \alpha$ where v is either a simple variable or an aggregate variable ; and (iii) nothing else, except the expressions obtained by finite applications of (i) and (ii), is in $Form(L_\Sigma)$. The definable syntactic objects $\cup$, $\cap$, $\leftrightarrows$, and $\exists$, and the standard notions such as *sentences* are also introduced in the usual way.

The definition of a well-formed formula above guarantees the unique readability of a formula given α. If α involves many parentheses, sometimes it is possible to omit certain parentheses in a formula without introducing any ambiguity. By adopting a standard convention of precedence between logical connectives, some parentheses will often be left out at our convenience. The logical connectives fall into three groups; $\neg$, $\cap$ and $\cup$, and $\rightarrow$ and $\leftrightarrows$, each of which is considered more binding than the one succeeding it. For example, according to the convention, $((\neg \phi \cap \psi) \rightarrow (\xi \cup \phi))$ can be written as $\neg \phi \cap \psi \rightarrow \xi \cup \phi$ unambiguously.

A few more notions of a well-formed formula such as the *existential quantifier* $\exists$ (defined as $\neg \forall \neg$), the *scope* of the quantifiers, the *bounded* variable, the *free* variable, and the *sentence* of L_Σ are adopted without stating their definitions explicitly.

3.2. Interpretation of L_Σ

A structure is needed to interpret each formula in L_Σ. A many-sorted structure for L_Σ, denoted by MS_a, consists of: (1) $|I|$ nonempty sets of objects $D_1, \cdots, D_{|I|}$ where D_i, $1 \leq i \leq |I|$, is called the domain of sort i of MS_a; (2) for each constant symbol $c \in C^i$, $1 \leq i \leq |I|$, an element $c^{MS_a} \in D_i$; (3) for each predicate symbol $R_k \in R^{<i_1, \cdots, i_n>}$, $\{i_1, \cdots, i_n\} \subseteq I$, a relation $R_k^{MS_a} \subseteq D_{i_1} \times \cdots \times D_{i_n}$; (4) for each function symbol $f \in F^{<i_1, \cdots, i_n, i_{n+1}>}$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, a function $f^{MS_a} : D_{i_1} \times \cdots \times D_{i_n} \rightarrow D_{i_{n+1}}$.

When it is convenient, MS_a is denoted by a quintuple $MS_a = < \{D_i\}_{i \in I}, P, R, F, C >$† where $\{D_i\}_{i \in I}$ is a sort domain set, P is a unary relation set whose members exclusively designate the ranges of aggregate variables, R is a relation set, F is a function set and C is a constant set.

The interpretation of a formula in the structure MS_a requires a variable assignment function s as follows:

<u>Definition 3.2.1</u>

For the set V of variables of L_Σ and the sort domain set $\{D_i\}_{i \in I}$ of structure MS_a, s is an assignment function, $s : V \rightarrow \bigcup_i D_i$, such that for a simple variable x_i of sort i, $s(x_i) = a$, where $a \in D_i$; and for an aggregate variable $x_i^{\Sigma Q}$ of sort i, $s(x_i^{\Sigma Q}) = a$, where if Q^{MS_a} ($Q^{MS_a} \subseteq D_i$) is the unary relation intended by Q in MS_a, then $a \in Q^{MS_a}$. ●

---

† To distinguish the elements of MS_a from those of L_Σ, usually a superscript is used such as R^{MS_a} or F^{MS_a}. However, the superscript MS_a is omitted if the distinction remains clear in the context.

Assignment function for the terms of L_Σ is defined as usual. For notational convenience symbol s is also used for the assignment for the terms. The validity of each formula is determined by the following interpretation rules.

Definition 3.2.2

For $A(t_1, \cdots, t_n) \in Atom(L_\Sigma)$, where A is an n-place predicate symbol of sort $<i_1, \cdots, i_n>$ and t_i's are terms, and $\psi, \psi_1, \psi_2 \in Form(L_\Sigma)$, the satisfaction of the formulas with respect to s in MS_a is defined by,

(1) $\models_{MS_a} A(t_1, \cdots, t_n)[s]$ iff $<s(t_1), \cdots, s(t_n)> \in A$,

(2) $\models_{MS_a} \neg \psi [s]$ iff $\not\models_{MS_a} \psi [s]$,

(3) $\models_{MS_a} \psi_1 \rightarrow \psi_2 [s]$ iff if $\models_{MS_a} \psi_1 [s]$ then $\models_{MS_a} \psi_2 [s]$,

(4) for a simple variable x_i of sort i , $\models_{MS_a} \forall x_i \; \psi [s]$ iff for any $a \in D_i$, $\models_{MS_a} \psi [s(x_i \mid a)]$,

(5) for an aggregate variable $x_i^{\Sigma Q}$ of sort i , $\models_{MS_a} \forall x_i^{\Sigma Q} \psi [s]$ iff for any $a \in Q^{MS_a}$, $\models_{MS_a} \psi [s(x_i^{\Sigma Q} \mid a)]$,

where for variables v_m and v_k , $s(v_m \mid a)(v_k) = \begin{cases} s(v_k) & \text{if } v_m \neq v_k \\ a & \text{if } v_m = v_k \end{cases}$ ●

As a corollary to the definition, the interpretations of $\cup$, $\cap$, $\leftrightarrows$, and $\exists$ can also be easily defined. In the following it is only shown how the existential quantifier $\exists$ is interpreted.

Lemma 3.2.1

For an existentially quantified formula $\exists v_i\ \psi$,

(1) If v_i is a simple variable x_i of sort i , $\models_{MS_e} \exists x_i\ \psi\,[s]$ iff for some $a \in D_i$, $\models_{MS_e} \psi\,[s(x_i \mid a)]$.

(2) If v_i is an aggregate variable $x_i^{\Sigma Q}$ of sort i , $\models_{MS_e} \exists x_i^{\Sigma Q}\ \psi\,[s]$ iff for some $a \in Q^{MS_e}$, $\models_{MS_e} \psi\,[s(x_i^{\Sigma Q} \mid a)]$.

Proof. First, it is remarked that $\exists v_i\ \psi$ is by definition $\neg\ \forall v_i\ \neg\ \psi$.

(1) If v_i is a simple variable x_i of sort i ,

$\models_{MS_e} \exists x_i\ \psi\,[s]$ iff it not the case that for all $a \in D_i$, $\not\models_{MS_e} \psi\,[s(x_i \mid a)]$

iff for some $a \in D_i$, $\models_{MS_e} \psi\,[s(x_i \mid a)]$.

(2) If v_i is an aggregate variable $x_i^{\Sigma Q}$ of sort i ,

$\models_{MS_e} \exists x_i^{\Sigma Q}\ \psi\,[s]$ iff it not the case that $\models_{MS_e} \forall x_i^{\Sigma Q} \neg\ \psi\,[s]$

iff it not the case that for all $a \in Q^{MS_e}$,

$\models_{MS_e} \neg\ \psi\,[s(x_i^{\Sigma Q} \mid a)]$

iff it not the case that for all $a \in Q^{MS_e}$,

$\not\models_{MS_e} \psi\,[s(x_i^{\Sigma Q} \mid a)]$

iff for some $a \in Q^{MS_e}$,

$\models_{MS_e} \psi\,[s(x_i^{\Sigma Q} \mid a)]$. *Q.E.D.*

3.3. Σ-Extensibility of L_Σ

As it should be clear now, the difference between L_Σ and L_m is that in L_Σ a new syntactic object, called an aggregate variable, is additionally featured. Introducing an aggregate variable in L_Σ is often different from introducing an ordinary sort variable in L_m for the reason that an aggregate variable's range is restricted to a unary relation instead of a sort domain. Let a unary predicate Q, for instance, be not in the alphabet of L_Σ. In such a case, the aggregate variables accompanying Q, for example $v^{\Sigma Q}$, may not be used in any formula of L_Σ. However, the fact that an aggregate variable's range is determined by the accompanying unary predicate implies that those aggregate variables accompanying Q, such as $v^{\Sigma Q}$, can be introduced if L_Σ is extended with the unary predicate Q. In this section, the process of introducing the aggregate variables whose accompanying unary predicates are not in L_Σ is formally described. The correctness of such process is also shown.

Let T_Σ be a theory in L_Σ and let $\sigma_\Sigma \in T_\Sigma$ be of the following form:

$$\sigma_\Sigma = \forall y_i \;\; (\alpha(y_i) \rightarrow \_\_\_\_\_\_\_ \;\; y_i \;\; \_\_\_\_\_\_\_) . \qquad (3.1)$$

Assuming that $\alpha(x)$ is a complex formula with x being a free variable, let Q be a defined symbol such as $Q(x) \leftrightarrows \alpha(x)$ so that σ_Σ of (3.1) can be equivalently expressed in a more compact form, say σ_Σ^e, as follows:

$$\sigma_\Sigma^e = \forall x_i \;\; (Q(x_i) \rightarrow \_\_\_\_\_\_\_ \;\; x_i \;\; \_\_\_\_\_\_\_) . \qquad (3.2)$$

The preceding way of abbreviating the formula σ_Σ is not satisfactory in the sense that Q in σ_Σ^e is not a predicate symbol at all. A more satisfactory way of doing this is to form what is called an extension of the theory T_Σ. The first step of this theory extension procedure is to augment L_Σ by a predicate symbol Q and to specify the meaning of Q in the form of

$$\forall x\,(Q\,(x) \leftrightarrows \alpha(x)) \tag{3.3}$$

where $\alpha(x)$ is a formula in L_Σ that does not contain the predicate symbol Q . The formula of (3.3) is called the defining axiom of Q . The second step is then to augment T_Σ by the abbreviated form σ_Σ^a in (3.2) as well as the defining axiom of Q in (3.3). It is clear that in the extended theory σ_Σ of (3.1) can be replaced by the abbreviated form σ_Σ^a of (3.2).

The preceding theory expansion procedure is described more specifically. Let L_Σ be $L_\Sigma = < P\ , R\ , F\ , C\ \ , \rho >$. Let P of L_Σ be augmented by a set P_Δ of new unary predicates symbols. The resulting language, denoted by L_Σ' , is formally called *a* Σ*-extension of* L_Σ . Let Δ be the set of unary predicate defining axioms of the form $\forall x\ (Q\,(x) \leftrightarrows \alpha(x))$ where $Q \in P_\Delta$ and $\alpha(x)$ contains only the unary predicate symbols in P or R of L_Σ . The theory T_Σ in L_Σ is augmented with Δ and in the augmented theory any formula of the form (3.1) is abbreviated to the formula of the form (3.3) by using respective defining axioms in Δ . The resulting theory, denoted by T_Σ' , is formally called *a* Σ*-extension of* T_Σ .

For the semantics of the new predicate symbols in a Σ-extended language L_Σ' of L_Σ , the unary relations corresponding to the newly introduced predicates must be introduced. If MS_a is a model of T_Σ , then it is not difficult to show that there is a unique expansion by definition of MS_a , say MS_a' , which is a model of T_Σ' . MS_a' is formally called *an expansion by* Σ*–definition* of MS_a . This process of expanding the structure for L_Σ is formalized by the following theorem.

Lemma 3.3.1

For a theory T_Σ in L_Σ, let MS_a be a model of T_Σ. If T_Σ' is a Σ-extension of T_Σ, then there is a unique expansion by Σ-definition MS_a' of MS_a which is a model of T_Σ' .

Proof. Let T_Σ' have been obtained from T_Σ by adding Δ defining axioms and abbreviating relativized expressions appropriately by using the respective defining axioms of Δ. Let L_Σ be $L_\Sigma = < P , R , F , C , \rho >$. Without loss of generality, it can be said that for some $n > 0$ Δ has been constructed in the following way: (i) Δ^0 is the set of all defining axioms of the form $\forall x\ (Q(x) \leftrightarrows \alpha(x))$ where $\alpha(x)$ contains only the predicates in P or R ; (ii) for any $j > 0$, Δ^j is the union of Δ^{j-1} and the set of all defining axioms of the form $\forall x\ (Q(x) \leftrightarrows \alpha(x))$ where $\alpha(x)$ contains only the unary predicates introduced in Δ^{j-1} or the predicates in P or R of L_Σ; (iii) then, for some $n > 0$, $\Delta \stackrel{d}{=} \Delta^n$.

It suffices to show that when T_Σ' is obtained from T_Σ by adding Δ a model of T_Σ' is obtained from MS_a by expansion by Σ-definitions and the model obtained in that way is unique. Let $\mathbf{\Delta}$ be $\mathbf{\Delta} = \{ \Delta^j : j = 1 , 2 , \cdots n \}$. Then it is noticed that there exists the partial ordering called " is a subset of " in $\mathbf{\Delta}$. The proof is shown by induction on the ordering of $\mathbf{\Delta}$.

For $j = 0$, let T_Σ^0 be obtained from T_Σ by adding the set Δ^0 of unary predicate defining axioms to T_Σ and modifying relativized expressions appropriately by using Δ. Let MS_a be $MS_a = < \{D_i\}_{i \in I} , P , R , F , C >$. For each defining axiom $\psi' \in \Delta^0$ which is of the form $\psi' = \forall x\ (Q(x) \leftrightarrows \alpha(x))$, let P of MS_a be augmented by the unary relation defined by $\alpha(x)$ in MS_a , i.e.,

$$\{ a : \models_{MS_a} \alpha(x)[a] \} \in P \quad \cdots (1).$$

Let the augmented structure be denoted by MS_a^0 , and let the defined relation of (1) be the interpretation of the predicate symbol Q in MS_a^0 . It must be shown that MS_a^0 is a model of T_Σ^0 and MS_a^0 is unique as a one that is obtained in the preceding way. It is first shown that MS_a^0 is a model of T_Σ^0 . For each formula $\psi' \in \Delta^0$, from the way that MS_a^0 was constructed it trivially follows that $\models_{MS_a^0} \psi'$. For each formula $\psi' \in T_\Sigma^0 - \Delta^0$, $\psi' \in T_\Sigma$ since T_Σ^0 is extended from T_Σ by Δ^0 . Since MS_a^0 is an expanded structure of MS_a and MS_a is a model of T_Σ , it holds that for each $\psi' \in T_\Sigma^0 - \Delta^0$, $\models_{MS_a'} \psi'$. Hence MS_a^0 is a model of T_Σ^0 .

Now the uniqueness of MS_a^0 is shown. Let MS_a^* be also a model of T_Σ^0 that is obtained from MS_a by expansion by Σ-definitions when T_Σ^0 is obtained from T_Σ by adding Δ^0 . Since MS_a^* is a model of T_Σ^0 , it should hold that for each defining axiom $\psi' \in \Delta^0$ $\models_{MS_a^*} \psi'$. This implies that for each unary predicate symbol introduced in Δ^0 its interpretation in MS_a^0 and MS_a^* are identical with each other. Since both MS_a^0 and MS_a^* are expanded from MS_a by using Δ^0 , the preceding result concludes that MS_a^0 and MS_a^* are identical.

For $j > 0$, it is assumed that when T_Σ^j is obtained from T_Σ^{j-1} by adding $\Delta^j - \Delta^{j-1}$ there is a unique expansion by Σ-definition MS_a^j of MS_a^{j-1} which is a model of T_Σ^j . The induction step is the following.

Let T_Σ^{j+1} be obtained from T_Σ^j by adding $\Delta^{j+1} - \Delta^j$. Let MS_a^j be

$MS_a^j = < \{D_i\}_{i \in I}, P^j, R, F, C >$. For each defining axiom $\psi' \in \Delta^{j+1} - \Delta^j$, say $\psi' = \forall x\ (Q(x) \leftrightarrows \alpha(x))$, let P^j be augmented by the unary relation defined by $\alpha(x)$ in MS_a, i.e.,

$$\{ a : \models_{MS_a} \alpha(x)[a] \} \in P^j \quad \cdots \ (2).$$

Let the augmented structure be denoted by MS_a^{j+1}, and let the defined unary relation of (2) be the interpretation of Q in MS_a^{j+1}. It can be shown that MS_a^{j+1} is a model of T_Σ^{j+1} in a way similar to the one that showed MS_a^0 is a model of T_Σ^0 for $j = 0$.

Showing the uniqueness of MS_a^{j+1} as a model of T_Σ^{j+1} is also similar to showing the uniqueness of MS_a^0 as a model of T_Σ^0. If MS_a^* is also a model of T_Σ^{j+1} that is obtained from T_Σ^j by adding $\Delta^{j+1} - \Delta^j$, then it can be shown that for each unary predicate introduced in $\Delta^{j+1} - \Delta^j$ its interpretations in MS_a^{j+1} and MS_a^* are identical with each other. Since both MS_a^{j+1} and MS_a^* are expanded from MS_a^j only by using the defining axioms in $\Delta^{j+1} - \Delta^j$, MS_a^{j+1} and MS_a^* must be identical. *Q.E.D.*

Now it is shown how a formula such as σ_Σ^o in (3.2) can be further abbreviated by introducing aggregate variables. Once the language L_Σ is extended with the unary predicate Q, by using the aggregate variable $x_i^{\Sigma Q}$ the formula σ_Σ^o in (3.2) can be syntactically translated into a more compact form, say σ_Σ', in the extended language L_Σ' of L_Σ as follows:

$$\sigma_\Sigma' = \forall x_i^{\Sigma Q} \ \_\_\_\_\_\_\_ \ x_i^{\Sigma Q} \ \_\_\_\_\_\_\_ \ . \qquad (3.4)$$

Let $\sigma_\Sigma{}'$ replace σ_Σ^o in T_Σ' . Overall, $\sigma_\Sigma{}' \in T_\Sigma'$ is derived from $\sigma_\Sigma \in T_\Sigma$ by introducing an aggregate variable along with L_Σ being extended to L_Σ' . This characteristic of L_Σ that allows a more compact expressive power in its extended language is called *Σ-extensibility* of L_Σ. In the rest of this section, the Σ-extensibility of L_Σ is justified, i.e., whether the procedure of translating $\sigma_\Sigma \in T_\Sigma$ in (3.1) into $\sigma_\Sigma{}' \in T_\Sigma'$ in (3.4) is correct or not.

One way of justifying the overall translation procedure is to show the following: for each formula $\psi' \in T_\Sigma'$ that contains some aggregate variable(s), say Λ , whose accompanying unary predicates are specified in a set Δ of defining axioms, there can be derived a formula in L_Σ which does not contain any aggregate variable in Λ and whose meaning is identical with that of ψ' .

Given a formula $\psi' \in T_\Sigma'$ that contains some aggregate variable(s), its translation process into L_Σ can be done exactly in the reverse way of what a formula such as σ_Σ in (3.1) was translated into $\sigma_\Sigma{}'$ in (3.4). The first step is to convert ψ' into its equivalent relativized expression in L_Σ' , i.e., translating a formula of the form (3.4) into a formula of the form (3.2). Let the relativized expression be denoted by ψ^o . The second step is to eliminate from ψ^o the newly introduced unary predicates by applying their respective predicate defining axioms, i.e., translating a formula of the form (3.2) into a formula of the form (3.1) by applying the defining axiom such as (3.3). The resulting formula of the second step does not contain any aggregate variable and is totally described in L_Σ. Formally stated, the resulting formula, say ψ^* , in L_Σ is called ***a translation of*** ψ' ***into*** T_Σ. The fact that ψ' and ψ^* have an identical meaning is shown by the following theorem:

Theorem 3.3.2

For a $\psi' \in T_\Sigma'$, if ψ^* is a translation of ψ' into T_Σ, then ψ' is true in MS_a' iff ψ^* is true in MS_a.

Proof. There are three kinds formulas in T_Σ' : defining axioms, formulas containing no aggregate variables and formulas containing aggregate variables. Here the formulas containing aggregate variables are only concerned. Let $\psi' \in T_\Sigma'$ be a formula that contains some aggregate variables. Proof is shown by induction on the length of ψ'.

First let ψ' be an atomic formula of the form $R(v_1', \cdots, v_n')$†, and let ψ' be true in MS_a' with an assignment function s' . Then the translation ψ^* of ψ' into T_Σ is done in the following way: if a variable v_i', $1 \leq i \leq n$, is an aggregate variable of the form $x_j^{\Sigma Q}$, then replace $x_j^{\Sigma Q}$ by x_j . Also along with such translation, an assignment function s for the variables in the translated formula is introduced in the following way: if v_i', $1 \leq i \leq n$, is an aggregate variable of the form $x_j^{\Sigma Q}$, then $s(x_j) = s'(x_j^{\Sigma Q})$, otherwise $s(v_i) = s'(v_i)$ [the assignment function s defined in the preceding is used throughout this proof]. Let the translation ψ^* that is obtained in the preceding way be $R(v_1, \cdots, v_n)$. Since $R^{MS_a'} = R^{MS_a}$, from the way s is defined it follows that

$$\begin{aligned} \models_{MS_a'} R(v_1', \cdots, v_n')[s'] &\iff < s'(v_1'), \cdots, s'(v_n') > \in R^{MS_a'} \\ &\iff < s(v_1), \cdots, s(v_n) > \in R^{MS_a} \\ &\iff \models_{MS_a} R(v_1, \cdots, v_n)[s]. \end{aligned}$$

† If A is a formula having free variables $v_1, \cdots, v_n$, then sometimes $A(v_1, \cdots, v_n)$ is written for A .

The theorem holds when ψ' is atomic.

Suppose that the result is true for all formulas of length less than or equal to h . It is shown that the inductive step holds for the formulas of length $h+1$. Let β' and γ' be formulas of length h , i.e., for their respective translations β^* and γ^* , let it hold that for any assignment function s' and its correspondingly defined assignment function s , $\models_{MS_e'} \beta'[s']$ iff $\models_{MS_e} \beta^*[s]$ and $\models_{MS_e'} \gamma'[s']$ iff $\models_{MS_e} \gamma^*[s]$. Inductive step is the following.

(Case I) Let ψ' be $\neg \beta'$. It follows that

$$
\begin{aligned}
\models_{MS_e'} \psi'[s'] &\iff \models_{MS_e'} \neg \beta'[s'] \\
&\iff \not\models_{MS_e'} \beta'[s'] \\
&\iff \not\models_{MS_e} \beta^*[s] \quad \text{by the induction hypothesis} \\
&\iff \models_{MS_e} \neg \beta^*[s] .
\end{aligned}
$$

Since ψ^* is $\neg \beta^*$, the theorem holds.

(Case II) Let ψ' be $\beta' \to \gamma'$. It follows that

$$
\begin{aligned}
\models_{MS_e'} \psi'[s'] &\iff \models_{MS_e'} (\beta' \to \gamma')[s'] \\
&\iff \text{if } \models_{MS_e'} \beta'[s'] \text{, then } \models_{MS_e'} \gamma'[s'] \\
&\iff \text{if } \models_{MS_e} \beta^*[s] \text{, then } \models_{MS_e} \gamma^*[s] \\
&\qquad \text{by the induction hypothesis} \\
&\iff \models_{MS_e} \beta^* \to \gamma^*[s] .
\end{aligned}
$$

Since ψ^* is $\beta^* \to \gamma^*$, the theorem holds.

(Case III) Let ψ' be $\forall x_i^j \beta'$ where x_i^j is a simple variable. It follows that

$$\models_{MS_e'} \psi' [s'] \iff \models_{MS_e'} \forall x_i^j \beta' [s']$$
$$\iff \text{for any } a \in D_i , \models_{MS_e'} \beta' [s' (x_i^j \mid a)]$$
$$\iff \text{for any } a \in D_i , \models_{MS_e} \beta [s (x_i^j \mid a)]$$
by the induction hypothesis
$$\iff \models_{MS_e} \forall x_i^j \beta [s]$$

Since ψ^* is $\forall x_i^j \beta$, the theorem holds.

(Case IV) Let ψ' be $\forall x_i^{\Sigma Q} \beta'$ where $\forall x\, (Q(x) \leftrightarrows \alpha(x)) \in T_\Sigma'$. It follows that

$$\models_{MS_e'} \psi' [s'] \iff \models_{MS_e'} \forall x_i^{\Sigma Q} \beta' [s']$$
$$\iff \text{for any } a \in Q^{MS_e'} , \models_{MS_e'} \beta' [s' (x_i^{\Sigma Q} \mid a)]$$
by $Q(x) \leftrightarrows \alpha(x)$ and $Q^{MS_e'} \subseteq D_i$
$$\iff \text{for any } a \in D_i , \text{ if } \models_{MS_e'} \alpha(x_i) [s' (x_i \mid a)] ,$$
$$\text{then } \models_{MS_e'} \beta' [s' (x_i \mid a)]$$
$$\iff \text{for any } a \in D_i , \text{ if } \models_{MS_e} \alpha(x_i) [s (x_i \mid a)] ,$$
$$\text{then } \models_{MS_e} \beta^* [s (x_i \mid a)]$$ by the induction hypothesis
$$\iff \text{for any } a \in D_i , \models_{MS_e} (\alpha(x_i) \rightarrow \beta^*) [s (x_i \mid a)]$$
$$\iff \models_{MS_e} \forall x_i (\alpha(x_i) \rightarrow \beta^*) [s (x_i \mid a)] .$$

Since ψ^* is $\forall x_i (\alpha(x_i) \rightarrow \beta^*)$, the theorem holds. *Q.E.D.*

The significance of the theorem is that passing from T_Σ to T_Σ' does not really do any more than express a formula in L_Σ more compactly by using aggregate variables. In addition to showing the compact expressive power of L_Σ , the above theorem suffices to justify the validity of embedding *aggregate variables* in a many-sorted language.

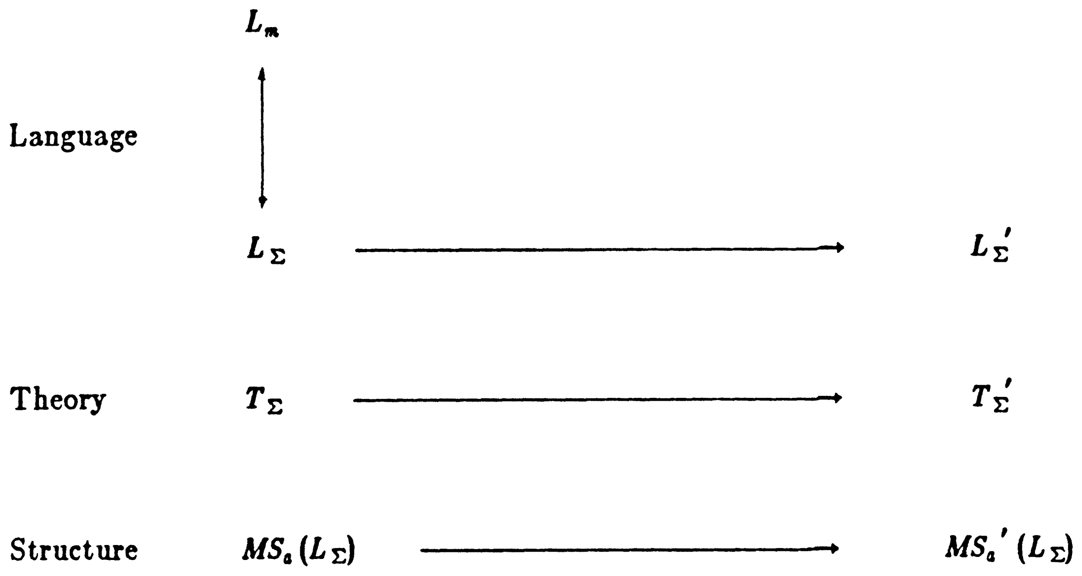

Figure 3.1. Summary of the Σ-extensibility of L_Σ

In conclusion, the advantage of L_Σ over L_m is that L_Σ offers a more compact expressive power than L_m . While the same advantage can be obtained in L_m if the sort structure of $MS(L_m)$ is changed, for L_Σ its associated structure $MS(L_\Sigma)$ only needs to be expanded by Σ-definitions. This characteristic has been called Σ-*extensibility* of L_Σ. The Σ-extensibility of L_Σ is summarized in Figure 3.1.

CHAPTER IV

PROBLEM FORMULATION

4.1. Modelling a Database and a Knowledge base

In this chapter, it is outlined how the knowledge-based approach described in Chapter II for the KBDDBS design is adopted by constructing a knowledge base system. As a preliminary step, in this section, the two notions, a database and a knowledge base associated with that database, are modeled in formal terms.

Let the terms by Gallaire and Minker be adopted who call data elementary facts of a real world and the knowledge about the data general facts of a real world [GaMi78]. In general, there have been two ways of formalizing the elementary facts of a real world and the general facts of the world. One way is to view both of them as a collection of homogeneous objects, i.e., a collection of sentences in some language. In this view, the collection of the sentences describing both of the elementary facts and the general facts in some language is regarded as a theory while the real world associated with both of the facts is regarded as a model of the theory. This view has been generally adopted in Q-A systems [Chan76, Mink78, Reit78b] in which both the elementary facts and the general facts are considered as sentences in some language so that the answers to the queries which go beyond the elementary facts can be derived from the general facts.

The other way is to view the two types of facts as two heterogeneous objects, i.e., the elementary facts as a logical structure and the general facts as a theory whose model is the logical structure. The second view better fits the context of database management systems(DBMS). In DBMS, a database is a collection of structured and formatted information which means a collection of elementary facts, while the integrity constraints are neither structured nor formatted information and are above the elementary facts, and, therefore, are regarded as general facts. Integrity constraints are to a database as a theory is to its model. That is, the validity of the integrity constraints has to be enforced within the database all the time, while the truthfulness of each sentence in a theory must be verified if the structure is to be a model of the theory. The elementary facts and the general facts in DBMS, may well be regarded as two different objects, one a logical structure and the other a theory.

In this work, this latter view is adopted since data and the knowledge about that data are considered as different categories of objects, i.e., data is a collection of relations to be distributed over a network whereas the knowledge about the data is used as a means of such distribution of the relations. The schematic representation of this view is shown in Figure 4.1.

A database is first formalized as a many-sorted structure. The intention is then to define a first-order language associated with the structure so that knowledge about the database and the user demands for the database can be described in this language. More specifically speaking, the first-order language defined† on the structure is a many-sorted language with aggregate variables L_Σ.

† Given any logical structure, a language associated with the structure can be easily defined. One procedure may be simply to collect all the symbols required and establish an interpretative connection between each symbol and each relation or function of the structure.

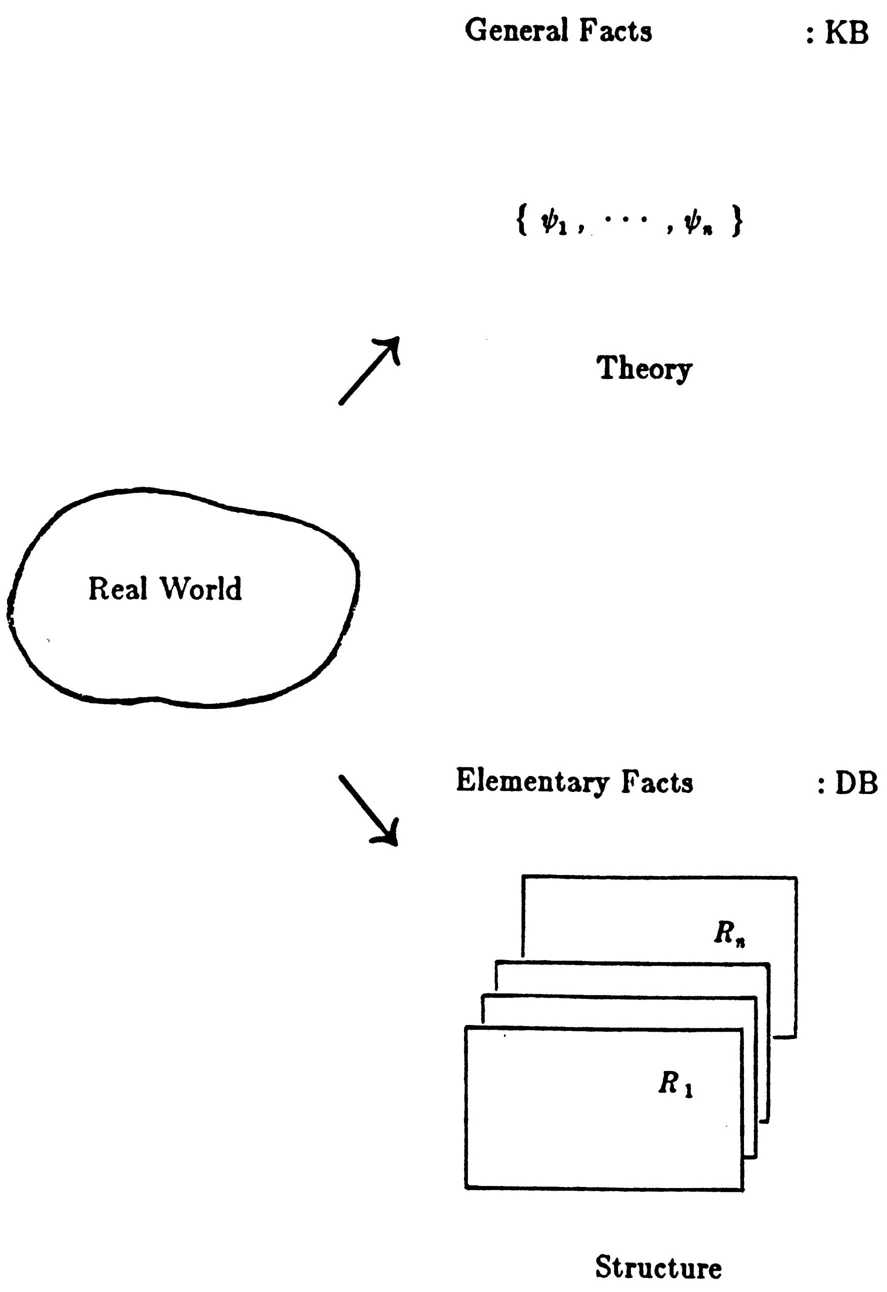

Figure 4.1. Modeling of a Database and a Knowledge Base

The database formalized here is a collection of relations to be distributed among the sites of a network. It is formally stated as follows:

Definition 4.1.1

A *database application*, denoted as DB, is an ordered structure $DB = < \{D_i\}_{i \in I}, \{P_i\}_{i \in I}, \{R_j\}_{j \in J}, \{C_k^i\}_{i \in I, k \in K_i} >$ with the associated function $\lambda : J \rightarrow N^+$ such that

(1) I is a domain index set where for each $i \in I$, D_i is the set of objects of i^{th} sort and each D_i constitutes the i^{th} universe of DB.

(2) J is a relation index set where for each $j \in J$, $\lambda(j)$ is a positive integer and R_j is $\lambda(j)$-ary relation on $\{D_i\}$, i.e.,

$$R_j \subseteq D_{i_1} \times \cdots \times D_{i_{\lambda(j)}} \quad i_n \in I .$$

(3) K_i is a (possibly empty) collection of constant names where for each $k \in K_i$, a distinguished element C_k^i is an element of D_i.

(4) P_i is a set of unary relations where if $p \in P_i$, then $p \subseteq D_i$. ●

In order to illustrate the preceding definition, a hypothetical distributed database is introduced. Throughout Part I, this database is used as the master example. Consider one big auto corporation which is going to develop a distributed database system. The corporation has part plants and assembly plants scattered around a large area with the headquarters located some distance from the plants. Suppose the corporation is planning to install three computing sites which will be connected via a computer network as shown in Figure 4.2, one in a parts plant(PP) complex, one in an assembly plant(AP) complex and one in headquarters(HQ). It is assumed that each computing site handles most of the transactions associated with it. That means, in PP the transactions of parts plants are handled, in AP the transactions of assembly plants, and in HQ the transactions typical of headquarters, for instance, the transactions of all the office items which are being consumed by the corporation.

Let { *DIVISIONS*, *DEALERS*, *ITEMS*, *SALES* } be a fraction of the database to be distributed over the three sites each of whose logical schemas is

DIVISIONS (*div#* , *div_name* , *head*)
DEALERS (*d#* , *address* , *d_type*)
ITEMS (*item#* , *i_name* , *i_type*)
SALES (*div#* , *d#* , *item#*)

where *div#* is a division number, *d#* a dealer number, and *item#* an item number. Each instance of these logical schemas is shown in Appendix A. In the following an example is illustrated showing how this database is formalized by a many-sorted structure. It is also illustrated how L_Σ is defined on the structure.

Example 4.1.1

Given the preceding auto corporation database, the many-sorted structure defined on this database, say *DB*(*Auto*), is $DB(Auto) = <\{item\#, d\#, \cdots\}, \phi,$ $\{DIVISIONS, DEALERS, ITEMS, SALES\}, \{B47, V02, \cdots\}>$, where *item#*, *d#*, $\cdots$ designate sort domains and $B47$, $V02, \cdots$ are constant symbols which are the members of *item#* sort domain. It is noticed that initially the unary relation set $\{P_i\}$ is empty.

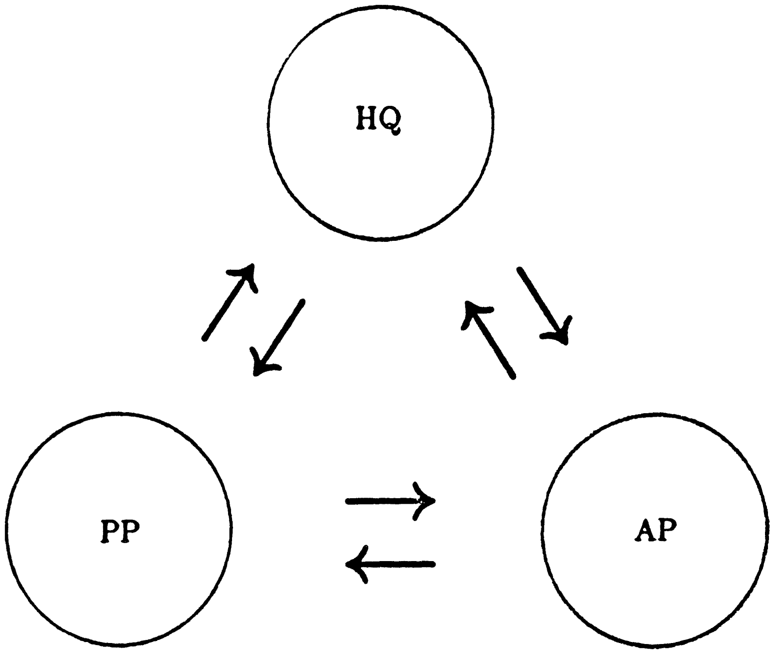

Figure 4.2. A Computer Network of an Auto Corporation

Now since there is no unary relation in the initially defined structure $DB(Auto)$, a L_{Σ} is introduced associated with the logical structure $DB(Auto)$ with its unary predicate set being empty. Let such L_{Σ} be denoted by $L_{\Sigma}^{\phi}(DB(Auto))$ to indicate that its unary relation set is empty. Assuming that all the symbols, needed for $L_{\Sigma}^{\phi}(DB(Auto))$ are provided, $L_{\Sigma}^{\phi}(DB(Auto)) = < \phi, \{ Div, De, It, Sa \}, \{ B47, V01, \cdots \}, \rho >$ where Div, De, It and Sa are the predicate symbols indicating the relations *DIVISIONS* , *DEALERS* , *ITEMS* and *SALES* , respectively; $B47$ and $V01$ are the constant symbols which belong to the *item#* sort domain, and; $\rho(It) = 3$ and so on. At this moment no unary predicate symbol has been yet introduced. However, it will be noticed later that

$L^{o}_{\Sigma}(DB(Auto))$ gradually evolves into $L_{\Sigma}(DB(Auto))$ as aggregate variables are introduced.

The preceding example shows how a database is formalized as a structure and a language is defined on the structure. Now a knowledge base associated with the database is formalized using the language defined on the structure, i.e., the knowledge base is a theory whose model is the structure.

Definition 4.1.2

For a database application DB, let $L_{\Sigma}(DB)$ be the many-sorted language with aggregate variables defined on DB . *A knowledge base associated with* DB, denoted by $KB(DB)$, is then a collection of some sentences of $L_{\Sigma}(DB)$ which are true in the structure DB . ●

The following illustrates the preceding way of formalizing the knowledge base about the data:

Example 4.1.2

Consider the language $L^{o}_{\Sigma}(DB(Auto))$ of Example 4.1.1. Let a general fact of the auto corporation world be expressed by a sentence ψ in $L^{o}_{\Sigma}(DB(Auto))$. As long as ψ describes a true fact of the real world which is modeled by $DB(Auto)$, ψ is interpreted as true in $DB(Auto)$. Therefore ψ is an element of the knowledge base associated with $DB(Auto)$, denoted by $KB(DB(Auto))$, i.e., $\psi \in KB(DB(Auto))$.

It is noticed that Definition 4.1.2 suggests an alternative way of defining a knowledge base. Suppose $Th(DB)$ means the complete theory defined on the structure DB, i.e., $Th(DB) = \{\psi : \models_{DB} \psi\}$. Then a knowledge base associated with DB is a subset of $Th(DB)$, i.e., $KB(DB) \subseteq Th(DB)$. In a practical environment, it is reasonable to assume that no complete theory can be defined on DB, or say a complete knowledge base. Only a necessary amount of knowledge which is useful for the intended purpose can be collected, so this can be at most a proper subset of the complete theory $Th(DB)$, if anything like $Th(DB)$ ever exists. Later in Chapter VI, it will be shown how the necessary amount of knowledge is gathered which constitutes the knowledge base intended to be built.

4.2. KBDDBS Design

In this section the KBDDBS design is formalized in terms of a function. By doing so the role of the knowledge base KB in the KBDDBS design is manifested. First, a local database in a network is defined as a logical structure. Then two DDBS design schemes, a DDBS design scheme which does not consider horizontal partitioning and the KBDDBS design scheme which does consider horizontal partitioning, is stated formally in terms of two different functions. By these two different functions, the differences between the two DDBS design schemes is visualized.

In a distributed environment, a local database is a fraction of the whole database which is a collection of relations. The fraction of the database may consist of fragments of relations if horizontal partitioning is adopted as design strategy, or complete copies of relations if horizontal partitioning is not considered. Defining a local database requires the notion of a *quasi-substructure* of a many-sorted structure to be introduced because the notion of *a substructure* is too restrictive to describe the local database containing fragments of relations.

The notion of a quasi-substructure is defined as follows: Let DB^a and DB^b be two database applications defined according to Definition 4.1.1, $DB^a = \{\{D_i^a\}_{i\in I}, \{P_i^a\}_{i\in I}, \{R_j^a\}_{j\in J}, \{C_k^{i,a}\}_{i\in I, k\in K_i}\}$ and $DB^b = \{\{D_i^b\}_{i\in I}, \{P_i^b\}_{i\in I}, \{R_j^b\}_{j\in J}, \{C_k^{i,b}\}_{i\in I, k\in K_i}\}$. Then DB^b is *a quasi-substructure of* DB^a, denoted by $DB^b \subseteq DB^a$, if the following conditions are satisfied:

(i) $D_i^b \subseteq D_i^a$, for each $i \in I$, and

(ii) $R_j^b \subseteq R_j^a \cap (D_{i_1}^b \times \cdots \times D_{i_{\lambda(j)}}^b)$ where $i_n \in I$.

The notion of a *quasi-substructure* is less restrictive than that of a *substructure* by the condition (ii). That is, for DB^b to be a substructure of DB^a, the condition (ii) should be $R_j^b = R_j^a \cap (D_{i_1}^b \times \cdots \times D_{i_{\lambda(j)}}^b)$ where $i_n \in I$.

Let L be a site index and $l \in L$. Then in terms of a quasi-substructure a local database application at a site l is formally stated as follows.

<u>Definition 4.2.1</u>

Let DB be the database application which is to be distributed over a network. For each $l \in L$, where L is an index set for sites, *a local database application of site* l, denoted as DB_l, is a quasi-substructure of DB, such as

$$DB_l = < \{D_{i,l}\}_{i\in I}, \{R_{j,l}\}_{j\in J}, \{C_{k,l}^i\}_{i\in I, k\in K_i} > \qquad \bullet$$

Example 4.2.1

Let $L=\{AP, PP, HQ\}$ and let the database to be dispersed be $DB(Auto)$ of Example 4.1.1. Suppose it has been decided to store in HQ only the information about the dealers which deal with office items. Let such information be represented by a subset of the relation *DEALERS*, called *OFFICE_DEALERS*, then the local database at HQ DB_{HQ} is

$$DB_{HQ} = <\{d\#, item\#, \cdots\}, \phi, \{OFFICE\_DEALERS, \cdots\}, \{ink, \cdots\}>$$

and $DB_{HQ}(Auto) \subseteq DB(Auto)$.

From the definition above, an allocation configuration of DB over a network could be simply formulated as a collection of quasi-substructures of DB, each of which is a local database. The only restriction on each allocation configuration of DB is that each tuple of each relation in the DB should reside in at least one site of the network. The notion of an allocation configuration of DB over a network is defined as follows:

Definition 4.2.2

Given a database application DB and a site index set L, *an allocation configuration of DB over L*, denoted by DDB, is a collection of quasi-substructures of DB such that if $DDB = \{DB_l : l \in L\}$ where $DB_l = < \{D_{i,l}\}_{i \in I}, \{R_{j,l}\}_{j \in J}, \{C^i_{k,l}\}_{i \in I, k \in K_i}>$, then $D_i = \bigcup_{l \in L} D_{i,l}$, $R_j = \bigcup_{l \in L} R_{j,l}$, and $C^i_k = \bigcup_{l \in L} C^i_{k,l}$. ●

In the following, two extreme cases of allocation configurations are shown, i.e., one an allocation scheme with complete redundancy and the other an allocation scheme without any redundancy.

Example 4.2.2

Let DDB_c and DDB_n be the allocation configurations with complete redundancy and without any redundancy, respectively. Let $DDB_c = \{DB_l^c : l \in L\}$, then $\forall l \in L \quad DB_l^c = DB$. Let $DDB_n = \{DB_l^n : l \in L\}$ where $DB_l^n = \ <\{D_{i,l}^n\}_{i \in I}$, $\{R_{j,l}^n\}_{j \in J}$, $\{C_{k,l}^{n}{}^i\}_{i \in I,\, k \in K_i} >$. Then $\forall j \in J \ \forall l_1, l_2 \in L$, if $l_1 \neq l_2$, then $R_{j,l_1}^n \cap R_{j,l_2}^n = \phi$

It is well known that there are two main DDB design criteria space cost and time cost, say C_s and C_t , respectively. Consider a DB and a network index set L . Let $D_{DB,L}$ stand for the collection of all the possible legitimate allocation configurations defined by Definition 4.2.1 including the two extreme cases of allocation configurations, the allocation scheme with complete redundancy and the allocation scheme without any redundancy. Let M be the index set of all the possible DDB 's, then

$$D_{DB,L} = \{DDB_m : m \in M\} .$$

A DDBS design scheme in general is then to produce the allocation configuration from the given DB and L , say $DDB_{min} \in D_{DB,L}$, in such a way that the costs C_s and C_t associated with DDB_{min} are less than any costs associated with each $DDB_m \in D_{DB,L}$.

The cost C_s associated with a DDB may be calculated by simply adding up all the storage costs required by the DDB. The cost C_t, however, can not be decided from an allocation configuration alone because calculating C_t usually requires a known or estimated system load as an additional parameter. In a distributed database system, such system load is usually modeled by a collection of ordered pairs $<$ a user query, the query issuance frequency $>$. Knowing how often each specific user query would be issued at each site is adequate information to judge a system's load.

In order to precisely define what a system load is, QF is introduced as a collection of ordered pairs of the following: $QF = \{<q_j^i, f_j^i> : q_j^i$ is the j^{th} query at the site i and f_j^i is the issuance frequency of $q_j^i\}$. As is implicit in the description of QF, QF depends on DB and L. Given a DB and L, if $QF_{DB,L}$ is the collection of all possible loads on the system of DB over L, then the two costs C_s and C_t are functions such as,

$$D_{DB,L} \xrightarrow{C_s} \$, \qquad D_{DB,L} \times QF_{DB,L} \xrightarrow{C_t} \$.$$

In the following, in terms of these two cost functions the two DDBS design schemes, a DDBS design without any partitioning of relations, and the KBDDBS design, are formalized.

<u>Definition 4.2.3</u>

Let S_{DB} and S_L be a set of various database applications DB's and a set of various site index sets L's, respectively. Given a $DB \in S_{DB}$ and a $L \in S_L$, *a DDBS design without any partitioning* is a function $f_{DB,L}^N : QF_{DB,L} \rightarrow D_{DB,L}$ such that for a $QF \in QF_{DB,L}$,

$$f_{DB,L}^{N}(QF) = DDB_{\min} ,$$

where if $DB = \{ \cdots ,\{R_j\}_{j \in J} , \cdots \}$, $DDB_{\min} = \{DB_l^m : l \in L\}$ and $DB_l^m = \{ \cdots ,\{R_{j,l}\}_{j \in J, l \in L} , \cdots \}$, then

(i) for any j and l , $R_{j,l} = \phi$ or $R_{j,l} = R_j$,

(ii) for each $DDB_i \in D_{DB,L}$ whose local database application does not allow any horizontally partitioned fragments of relations (i.e., if $DDB_i = \{DB_l^i : l \in L\}$ and $DB_l^i = \{ \cdots ,\{R_{j,l}^i\}_{j \in J, l \in L} , \cdots \}$, then for any j and l , either $R_{j,l}^i = \phi$ or $R_{j,l}^i = R_j$),

$$C_s(DDB_{\min}) + C_t(DDB_{\min}, QF) \leq C_s(DDB_i) + C_t(DDB_i, QF) .$$ •

<u>Definition 4.2.4</u>

Let S_{DB} and S_L be a set of various database applications DB's and a set of various site index sets L's, respectively. For a $DB \in S_{DB}$, suppose KB_{DB} is a set of all possible instances of a knowledge base associated with the database DB , i.e., the collection of all $KB(DB)$'s. Given a $DB \in S_{DB}$ and a $L \in S_L$, *the KBDDBS design* is a two place function $f_{DB,L}^{H} : QF_{DB,L} \times KB_{DB} \rightarrow D_{DB,L}$ such that for a $QF \in QF_{DB,L}$ and a $KB(DB) \in KB_{DB}$,

$$f_{DB,L}^{H}(QF, KB(DB)) = DDB_{\min}^{KB} ,$$

where if $DB = \{ \cdots ,\{R_j\}_{j \in J} , \cdots \}$, $DDB_{\min}^{KB} = \{DB_l^m : l \in L\}$, and $DB_l^m = \{ \cdots ,\{R_{j,l}\}_{j \in J, l \in L} , \cdots \}$, then

(i) for any j and l , $R_{j,l} \subseteq R_j$,

(ii) for each $DDB_i \in D_{DB,L}$,

$$C_s(DDB^{KB}_{\min}) + C_t(DDB^{KB}_{\min}, QF) \leq C_s(DDB_i) + C_t(DDB_i, QF) . \qquad \bullet$$

It is intuitively obvious that for a very large database, in which the queries at each site are more locally clustered at some fractions of the relations, the system designed by the design scheme of $f^H_{DB,L}$ would significantly outperform the system designed by the design scheme of $f^N_{DB,L}$. In other word, given DB, L , QF , and $KB(DB)$, if $f^N_{DB,L}(QF) = DDB_{\min}$ and $f^H_{DB,L}(QF, KB(DB)) = DDB^{KB}_{\min}$, then

$$C_s(DDB_{\min}) + C_t\ (DDB_{\min}, QF) \gg C_s(DDB^{KB}_{\min}) + C_t\ (DDB^{KB}_{\min}, QF) .$$

It is mainly because, in $f^N_{DB,L}$, though it depends on the system parameters and the degree of replication, the additional storage requirements plus the cost of maintaining consistent multiple copies of relations at more than one site all the time could be prohibitively high and, therefore, the cost paid for replication becomes more than the benefits gained by replication. This problem of $f^N_{DB,L}$ originated from viewing the whole body of each relation as the smallest unit of data allocation. In $f^H_{DB,L}$, however, the unit object of distribution is allowed to be the horizontally partitioned fragments of relations. That means, the storage and update cost for the unnecessarily replicated portion of relations may be avoided by appropriately distributing the horizontally partitioned fragments.

The major question in the KBDDBS design $f^H_{DB,L}$ is then "How should the relations be partitioned horizontally so that at each site the necessary portion or unnecessary portion of each relation can be faithfully reflected in their allocation?"

It is not difficult to see that the *URC*'s at each site are necessary to partition relations horizontally. As long as it is known that the user queries at a site are clustered around only a certain fraction of a relation, it would certainly be better to allocate only that fraction of the relation at the site.

Therefore, the problem of "How to partition relations horizontally?" is the problem of "How to identify the *URC*'s at each site?" One suggested approach would be to examine the user queries at each site. However, the approach to detect the *URC*'s precisely enough may not be feasible only by looking at the queries because the user queries are not formulated to do so. In the relational data model, users need not specify details about what they want from the database in their query expressions since the details about the database are transparent to the users. The information contained in the user queries is not sufficient to estimate the *URC*'s precisely.

Nevertheless, it is postulated that the DB designer can estimate the precise *URC*'s to some degree by exploiting the knowledge inherited from his(her) conception of the real world. After seeing a query, what the DB designer may do for this would be to search for any knowledge which may be applicable to the query and to derive better *URC*'s associated with the query. It is suggested that a knowledge-based approach can be employed in which what the DB designer does is mimiced. In the following section, the knowledge-based approach is discussed in detail.

4.3. Knowledge-Based Approach of the KBDDBS Design

In the previous section, it was discussed that what mattered in the KBDDBS design was to determine the *URC*'s precisely and it could be done by employing a knowledge-based approach. In this section it is discussed in detail how the knowledge-based approach is employed by constructing a knowledge base system.

The suggested approach is to build a front-end Knowledge-Based System (KBS) that receives the user provided queries, and revises them into equivalent versions by using the knowledge about the data which provide more precise *URC*'s than do the user provided queries. In building the knowledge-based system, there are two fundamental issues to be concerned with as in building any type of knowledge-based system: ***knowledge representation formalism*** and ***inference mechanism***. These two aspects of a knowledge-based system vary depending on a system's domain of applications. No general solution exists. However, one basic philosophy of a logical formal system may be applied in designing a knowledge based system, although a knowledge based system of AI differs from the formal system of logic from the practical point of view. The basic philosophy is stated in [Shoe67]:

> "Clearly whether or not A is a theorem of T depends strongly on what the nonlogical axioms of T are. Hence we must expect the condition for A to be a theorem of T to refer not only to A, but also to the nonlogical axioms of T. If these nonlogical axioms are sufficiently simple, this will not be a disadvantage. For theories with complicated nonlogical axioms, it is necessary to abandon [a] general solution, and seek a solution adapted to the particular theory."

If the knowledge base of any knowledge based system is regarded as a collection of nonlogical axioms with sufficient complexities, what is being implied by the preceding statement is that it would be desirable to develop a specific inference mechanism for a particular knowledge-based system rather than to develop a general inference mechanism applicable to any knowledge-based system. In this study, this philosophy is faithfully followed. When building a knowledge base for the KBDDBS design, the knowledge useful for its intended purpose is expressed in some specific types of formulas in L_{Σ} and an inference mechanism is developed which could be efficiently applied only to those formulas.

In the rest of this section, the KBS of the KBDDBS design is formalized in terms of a knowledge representation formalism and an inference mechanism. As a preliminary step, a knowledge-based system in general is formalized as follows:

Definition 4.3.1

A *knowledge-based system in general* (KBSG) is an ordered pair, $KBSG = \langle KRF, IM \rangle$, where KRF and IM stand for a knowledge representation formalism and an inference mechanism respectively, such that from the known facts expressed in KRF some additional true fact is efficiently deducible only by using the syntactical processing based on IM. ●

Example 4.3.1

If a propositional calculus (PC) in logic is considered as analogous to a knowledge based system in AI, then the collection of the tautologies and the nonlogical axioms of PC is considered to be a knowledge base and modus ponens with refutation procedure as an inference mechanism, therefore,

PC = < syntax rule of PC, modus ponens with refutation procedure >.

The practical systems in AI, the Q-A systems [Mink78, Reit78b] and QUIST system [King81] can be formalized into

Q-A = < syntax of applied first order language, resolution principle >,

QUIST = < syntax of QUIST query language, inference guiding heuristics > .

It becomes clear from the preceding examples, depending on the problem domain to which a knowledge based system is applied, *KRF* and *IM* of a *KBSG* vary. For instance, compare a Q-A system with the QUIST system. In a Q-A system a query is furnished as a conjectured theorem which ought to be proved, while in the QUIST system, a query is given and a set of equivalent queries may be derived. The former, therefore, naturally appeals to automatic theorem proving technique (ATP), which means *KRF* and *IM* of a Q-A system may be mapped into a formal language syntax and the refutation technique respectively. The latter, however, is not adequate for the application of ATP, because there is no conjectured theorem given a priori. Because of this reason and the fact that there may be many equivalent queries deducible which may not necessarily be beneficial, in QUIST a specific inference guiding heuristics has been chosen as its *IM* and at the same time *KRF* has been developed to fit its *IM*.

The KBS of the KBDDBS design is similar to the QUIST system in the sense that there is no conjectured theorem, but it differs from the QUIST system by the fact that there should be only one conclusion to be deduced by its inference mechanism. Suppose the KBS is modeled by

$$< \text{Syntax of } L_\Sigma ,\ IM^* >$$

where IM^* is some inference mechanism applicable to the formulas in L_Σ. In this case devising the inference mechanism IM^* to be applied to any formulas of L_Σ could be extremely difficult because L_Σ is a very general knowledge representation formalism. Fortunately, the fact that the useful knowledge for partitioning horizontally falls into a class of specific types of formulas in L_Σ allows the development of a simple efficient inference mechanism of the KBS. Following the philosophy of a logical formal system quoted earlier, it is suggested that the KBS be modeled by

$$< \Sigma\text{-Horn formula}\ ,\ \longmapsto >$$

where Σ-Horn formulas are some specific types of formulas in L_Σ and $\longmapsto$ is a simple syntactic matching procedure which is applicable only to the Σ-Horn formulas.

The schematic diagram of the horizontal partitioning system of the KBDDBS design is shown in Figure 4.3. The KBS consists mainly of two parts, namely, the knowledge base constituting some specific knowledge about the data, and the inference mechanism.

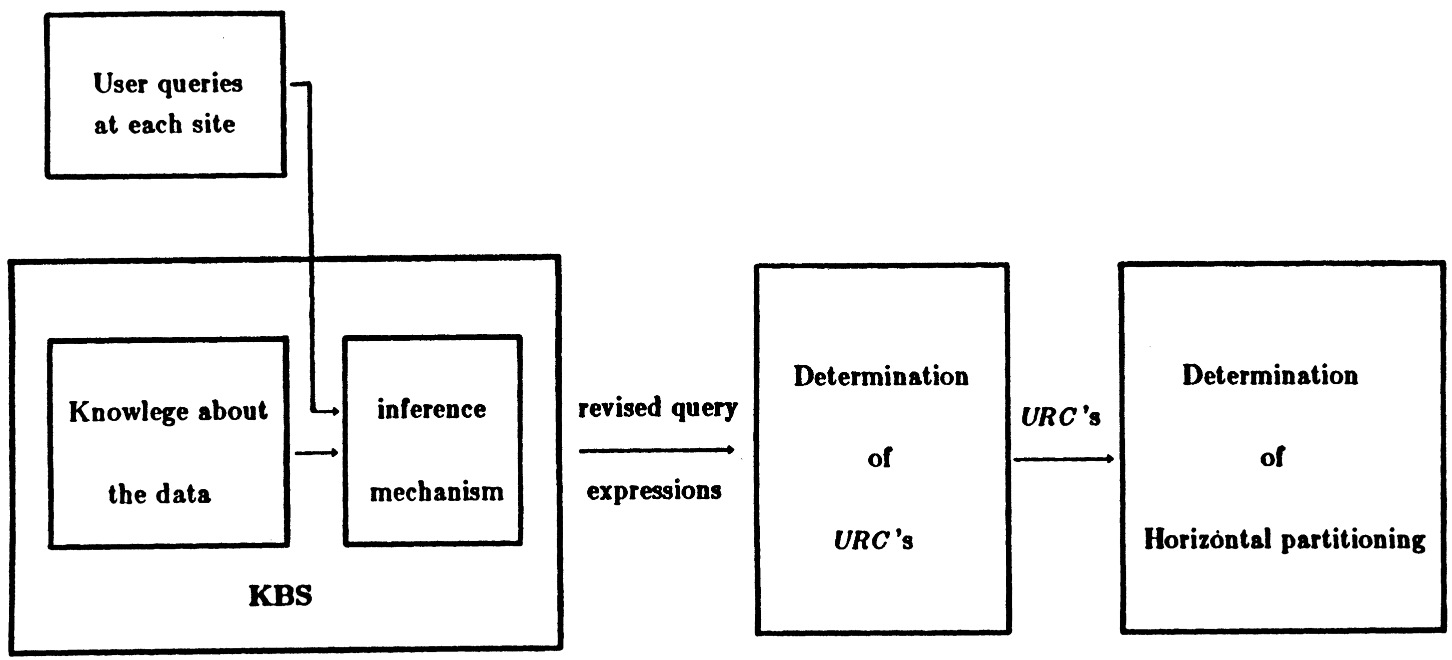

Figure 4.3. Horizontal Partitioning System of the KBDDBS design

CHAPTER V

QUERY REPRESENTATION IN L_Σ

5.1. Scheduled User Queries

User demand to the database, or simply saying user queries, is one of the fundamental design resources in most of the DDBS designs schemes. In this chapter it is shown how this fundamental design resource is expressed for the KBDDBS design by using L_Σ as the descriptive tool. First, in this section the notion of scheduled user queries is introduced and its importance is discussed.

When user queries are used for a DDBS design purpose, what information needs to be acquired from the user queries determines how the user queries should be expressed, i.e., the intended usage of the user queries determines query representation formalism. For example, in [Aper81], all the information needed from the queries is which kinds of relations appear in each query. Therefore, their query representation formalism only contains the information regarding what relations are needed to answer a query. Such representation formalism is sufficient for their intended purpose, since the issue of their study is to reflect the intermediate data flow only in terms of relations. Knowing which relations would be required to answer queries is enough to determine which distribution configuration of the relations would be the optimal.

However, in our study where the issue is to introduce a methodology of partitioning relations horizontally based on the *URC*'s, the *URC*'s obtained only in terms of the relations are not tight enough. The *URC*'s should be identified in terms of fractions of relations. In order to make it so, the query representation formalism in the KBDDBS design must contain the information regarding which fractions of relations are required to answer a query. Beside the preceding requirement, there is another condition to be satisfied for the query representation formalism in the KBDDBS design. That is, since the knowledge about the data is applied to a query, the query should be expressed in a compatible way with the knowledge to be applied. In summary, there are two issues involved in representing the user queries for the KBDDBS design:

(1) In the query expression, restrictions should be explicitly specified, as well as projections or joins, because the horizontal partitioning is mainly determined by restrictions.

(2) The queries should be expressed in a compatible way with the knowledge so that the knowledge can be applied to the queries via some syntactic inference process.

With these two issues in mind, the two notions, "user queries" and "scheduled user queries," are differentiated as follows. *User queries* are the instances of queries issued by the users which may be identified by a DB designer by intensive interviewing the users before attempting to design and *scheduled user queries* are the query expressions derived from the user queries in a "collective" way. What is meant by collective way is explained in the following.

First, let the notion of "same type" of user queries be defined as follows: For a given set of user queries, if any pair of user queries differ only by the values of restriction, then the queries in the set are of *the same type*. A scheduled query representing the set of the user queries of the same type is then any single formal expression that has the meaning of combining all the user queries in the set. For instance, suppose there are two user queries such as

"Who has been supplied item#=B47?"

"Who has been supplied item#=V03?"

These two queries are of the same type since they only differ by the restriction values B47 and V03. A scheduled user query derived from these two user queries is any formal expression having the same meaning as

"Who has been supplied item#=B47 or V03?" (5.1)

Regarding the first issue, i.e., restrictions should be explicitly specified in the query expression, a scheduled user query certainly contains the information regarding the subsets of relations on which restrictions are made. For example, any formal expression for (5.1) can indicate that the restrictions of the user queries are only to the transactions whose item# values are $B47$ or $V03$.

Now for the second issue, i.e., the queries should be expressed in a compatible way with the knowledge about the data, it is suggested that L_Σ be used as the representational tool of the scheduled user queries. By using aggregate variables in L_Σ, the collection of the same type of user queries can be compactly expressed in L_Σ. How scheduled user queries are expressed compactly in L_Σ is the content of the following section.

5.2. Σ-Normal Form as a Query Representation Formalism

There have been many languages suggested, and implemented in practice, as tools for representing queries. Each query language has been developed for its own purpose and these languages are compared to one another on the basis of different criteria. When concern is only with a relational data model, the query languages are divided into two types: algebraic languages and predicate calculus languages. The calculus-based languages are further divided into two classes, namely, tuple relational calculus and domain relational calculus. The primitive objects of the former are tuples of relations and those of the latter are elements of the domain of the same attributes. Here L_Σ is used as a query language based on the domain relational calculus.

In general, a query expression means the set of tuples to be returned as the answer to the query. Either in an algebraic query language or in a calculus-based query language, the syntax rules for query expressions are made up so that a query expression written according to those rules is intended to mean the set of tuples returned as the answer.

Suppose there are two relations R_i and R_j each of whose arity is two and which are joinable via their key attributes. If a query q is the one retrieving tuples of R_i whose key attribute is the same as the one of R_j, then in an algebraic language q would be expressed as

$$R_i \rhd\!\!\lhd R_j \ .$$

Compared to this, in a calculus-based language q would be expressed as

$$q = \{ <x,y> : R_i(x,y) \cap R_j(x,z) \} .$$

It is clear in the above example that unlike an algebraic language, a calculus-based language allows the query expression to be built up in two layers: one the intentional qualification clause which is a well-formed-formula of the calculus; and the other the bracket "{ }" intended to mean the set implied by the qualification clause. The well-formed-formula is called *a qualification clause* and the complete representation of a query which is intended to mean the answer set of the query is called *a query expression*. When representing the queries in L_Σ which is a domain relational calculus, a query's qualification clause must also be explicitly distinguished from its expression. The separation of the two notions is essential in this study since the knowledge is not applied to the query expression but to the qualification clause of the query.

In order to separate the two notions in the context of L_Σ, the notion of $DEF(\cdot)$ is first introduced as follows:

Definition 5.2.1

Given a structure DB, let $L_\Sigma(DB)$ be a language associated with DB. If ψ is a formula in $L_\Sigma(DB)$ with n free variables, then $DEF(DB,\psi)$ is an n-ary relation such that

$$DEF(DB,\psi) = \{<a_1, \cdots, a_n>: \models_{DB} \psi[s]\},$$

where if V is the variable set of $L_\Sigma(DB)$ and $\{D_i\}$ is the set of sort domains of DB, then s is a variable assignment function $s : V \to \bigcup_{i \in I} D_i$.

●

Now the two notions, a query expression and a query clause, are formally introduced as follows. Given a database application structure DB, let $\psi(v_1, \cdots, v_n) \in Form(L_\Sigma(DB))$. Then *a query expression* q is an expression of the form $DEF(DB, \psi(v_1, \cdots, v_n))$ and *the query clause of* q is $\psi(v_1, \cdots, v_n)$ of $DEF(DB, \psi(v_1, \cdots, v_n))$.

Example 5.2.1

Suppose the following is a user query to $DB(Auto)$ frequently issued at site AP: "What are the addresses of the dealers who were supplied item#=B47?" Then the expression of this query in L_Σ is,

$$q = DEF(DB(Auto), \psi_1),$$

where the query clause ψ_1 of q is

$$\psi_1 = \exists x \; \exists y \; \exists v \; (Sa(x, y, B47) \cap De(y, u, v)).$$

As long as the notion of $DEF(\cdot)$ is clear, from here on by simply a query it would be often meant a query clause. In the rest of the section it is shown that query clauses for scheduled user queries are of certain form in L_Σ. As stated in the previous section, an issue of how the queries should be expressed is whether the user queries can be expressed in a compatible way so that the knowledge can be applied to the user queries in a deductive way. With this in mind, a class of formulas of L_Σ is defined as follows:

Definition 5.2.2

For $\alpha \in Form(L_{\Sigma})$, α is in *Σ-normal form* if α is of the form, for some $n \geq 0$, $\exists v_{i_1} \cdots \exists v_{i_m} \psi(v_1, \cdots, v_n)$, where $\{v_{i_1} \cdots v_{i_m}\} \subset \{v_1, \cdots, v_n\}$, such that

(1) v_i, $1 \leq i \leq n$, is a simple or an aggregate variable, and

(2) $\psi(v_1, \cdots, v_n)$ is a conjunction of atomic formulas. ●

From here on, the Σ-normal form is used to express the query clauses for scheduled user queries. The expressive power of Σ-normal form is illustrated by an example.

Example 5.2.2

In addition to the user query clause ψ_1 shown in Example 5.2.1, suppose there are other user queries, say ψ_2 and ψ_3, as follows:

$$\psi_2 = \exists x \; \exists y \; \exists v \; (Sa(x, y, V01) \cap De(y, u, v)), \text{ and}$$

$$\psi_3 = \exists x \; \exists y \; \exists v \; (Sa(x, y, V03) \cap De(y, u, v)).$$

Then ψ_1, ψ_2, and ψ_3 are all of the same type. The scheduled user queries made up of ψ_1, ψ_2, and ψ_3 is a query expression asking "What are the addresses of the dealers which were supplied item# = $B47$, $V01$, or $V03$?" Without using aggregate variables, one way to express the query is $DEF(DB(Auto), \psi_1 \cup \psi_2 \cup \psi_3)$. In fact, the disjunctive form $\psi_1 \cup \psi_2 \cup \psi_3$ can be equivalently expressed as

$$\psi_1 \cup \psi_2 \cup \psi_3 = \exists x \, \exists y \, \exists v \, \exists z \, (Sa(x,y,V03) \cap De(y,u,v) \cap (z=B47 \cup z=V01 \cup z=V03)) .$$

Now it is shown how this scheduled user query can be compactly expressed in Σ-normal form. Suppose $L_\Sigma(DB(Auto))$ is Σ-extended by a new predicate symbol J and at the same time the structure $DB(Auto)$ is also expanded by Σ-definition by the defining axiom of J such as

$$\forall x \, (J(x) \leftrightarrows (x=B47 \cup x=V01 \cup x=V03)) .$$

Then by introducing an aggregate variable, the disjunctively conjoined formula $\psi_1 \cup \psi_2 \cup \psi_3$ collapses into a Σ-normal form formula q_1 as follows:

$$q_1 = \exists x \, \exists y \, \exists z^{\Sigma J} \, \exists v \, (Sa(x,y,z^{\Sigma J}) \cap De(y,u,v)) . \quad (5.2)$$

Here the scheduled user query expression $DEF(DB(Auto), \psi_1 \cup \psi_2 \cup \psi_3)$ is equivalently expressed by $DEF(DB(Auto), q_1)$.

•

In the preceding example, it is clear that q_1 of (5.2) is much more compact than the disjunctively conjoined formula $\psi_1 \cup \psi_2 \cup \psi_3$. User queries are expressed in a much more compact way in L_Σ than in an ordinary many-sorted language. How such compact way of expressing the query allows the application of the knowledge to the query is discussed in detail in Chapter VII. From here on as long as the distinction between user queries and scheduled user queries is clear, i.e., the latter is made from the former to be used for the purpose of the KBDDBS design, by simply "queries" it is meant scheduled user queries.

Finally, by using the formality of the query clauses in Σ-normal form, the notion of the *URC*'s which has been introduced informally in Chapter II is defined in terms of the atomic formulas of L_Σ as follows:

Definition 5.2.3

For a query q in Σ-normal form, let the matrix of q be of the form $R_1 \cap \cdots \cap R_m$ where $R_i \in Atom(L_\Sigma)$, $1 \leq i \leq m$. If $DEF^i(DB, q)$ stands for the singleton set whose member is the set $DEF(DB, R_i)$, then *the URC identified by* q is

$$\bigcup_{i \in \{1, \cdots, m\}} DEF^i(DB, q)$$ ●

An example of the preceding definition follows:

Example 5.2.3

The query q_1 of (5.2) in Example 5.2.2 is considered. The matrix of q is $Sa(x, y, z^{\Sigma I}) \cap De(y, u, v)$. The *URC*'s identified by q_1 is then the set

$$\{ DEF(DB(Auto), Sa(x, y, z^{\Sigma I})), DEF(DB(Auto), De(y, u, v)) \} .$$

CHAPTER VI

KNOWLEDGE REPRESENTATION IN L_Σ

6.1. Axiomatic Knowledge Identification

In this section it is discussed what kind of knowledge is included in the knowledge base of the KBS. In any knowledge-based system, what types of knowledge should be included in its knowledge base generally depends on the purpose of using the knowledge. In the KBDDBS design, the purpose of using the knowledge about the data is for the horizontal partitioning, and by doing so to eventually reap the benefits accrued from allocating the partitioned fragments instead of the complete relations. The benefits accrued when distributing horizontally partitioned fragments include: (i) during the process of queries, the selection operation is dispensed with in some degree by presuming each fragment as a preselected subrelation, and (ii) the unnecessary join operations are eliminated by knowing a priori the fact that join operations between some fragments produce a null set.

Such benefits, which are sought by relying on the dispersion of horizontally partitioned relations, imply what should be derived from the knowledge base and, therefore, what should be in the knowledge base. They are mainly the two types of knowledge: (i) the knowledge which contains the notion of preselection (or, say, prepartitioned fragments) which would be of benefit to dispense with the selection operations of queries, and (ii) the knowledge which shows the relationships between the prepartitioned fragments of relations which would eliminate any unfruitful join operations.

It is postulated that these two types of knowledge are expressible in terms of five types of axiom schemas in L_Σ. In other words, the instances (i.e., axioms) of five axiom schemas constitute the knowledge base of the KBS which is utilized for the purpose of horizontal partitioning of relations.

The five axiom schemas identified are Functional Dependency Axiom schema (FDA), Relationship Axiom schema (RA), Inherency Axiom schema (IA), Ground Defining Axiom schema (GDA) and Virtual Defining Axiom schema (VDA). The reason the knowledge is classified into the axioms of five types is twofold. One is to identify the knowledge useful for horizontal partitioning via syntactic formality, and the other is to exploit the formality for developing an inference mechanism. In the following, the meaning of each axiom schema is first briefly explained and then the representation of the schema in L_Σ is shown. Examples of each schema are given in the following section. These examples are annotated at each schema description. To simplify the expressions, some abbreviations are adopted: if A is an index set such as $A=\{a_1, \cdots, a_n\}$, then $\overline{X}_A$ and $Q\overline{X}_A$ are the abbreviations of the sequences $x_{a_1}, \cdots, x_{a_n}$ and $Qx_{a_1} \cdots Qx_{a_n}$, respectively, where Q is either $\forall$ or $\exists$.

(i) Functional Dependency Axiom (FDA)

Functional dependency (FD) in a relation is a well known concept. Any type of FD can be expressed in the form of schema discussed below and also any axiom of this schema describes a FD (e.g., (6.3)).

FDA schema : Given an n-ary relation R whose attribute index set is $\{1, \cdots, n\}$, if there is a FD from $\overline{X}_A$ to $\overline{X}_B$ where A and B are the subsets of $\{1, \cdots, n\}$, and $A \cap B$ is not necessarily the empty set, then the FD is expressed as $\forall \overline{X}_A \ \forall \overline{X}_{B'} \ \forall \overline{X}_C \ \forall \overline{Y}_{B'} \ \forall \overline{Y}_C \ (R(\overline{X}_A, \overline{X}_{B'}, \overline{X}_C) \cap R(\overline{X}_A, \overline{Y}_{B'}, \overline{Y}_C) \rightarrow (x_{1'} = y_{1'} \cap \cdots \cap x_{n'} = y_{n'}))$ where all the variables of $\overline{X}_A$, $\overline{X}_{B'}$, $\overline{X}_C$, $\overline{Y}_{B'}$, and $\overline{Y}_C$ are simple variables, and $B' = B - A$, $C = \{1, \cdots, n\} - (A \cup B)$, and each i' is an element of B'.

(ii) Relationship Axiom (RA)

An axiom of RA schema describes the following types of relationships which hold between two relations: (i) whether an attribute of one relation shares a common domain with an attribute of the other relation, and (ii) if so, whether join of the two relations over the attributes of the common domain is meaningful in the sense that queries including the join of the two relations on these attributes are meaningful. Although any two relations having attributes which share a common domain are actually joinable, not every join of such relations would be meaningful. Only meaningful join of two relations is specified in this schema (e.g., (6.10)).

RA schema : Given an n-ary relation R_1 and an m-ary relation R_2 whose attribute index sets are $\{1, \cdots, n\}$ and $\{1, \cdots, m\}$ respectively, if the range of x_i, $i \in \{1, \cdots, n\}$, is identical with the range of y_j, $j \in \{1, \cdots, m\}$, and the join of R_1 and R_2 on x_i and y_j is meaningful, then such relationship between the two relations R_1 and R_2 is expressed as $\exists x_i \ \exists \overline{X}_{A'} \ \exists y_j \ \exists \overline{Y}_{B'} \ (R_1(x_i, \overline{X}_{A'}) \cap R_2(y_j, \overline{Y}_{B'}) \cap (x_i = y_j))$ where $A' = \{1, \cdots, n\} - \{i\}$, $B' = \{1, \cdots, m\} - \{j\}$ and all the variables x_i, $\overline{X}_{A'}$, y_j and $\overline{Y}_{B'}$ are simple variables.

(iii) Inherency Axiom (IA)

An axiom of IA schema is the type of knowledge which plays the key role in the KBS, since *URC*'s are more precisely estimated by knowing the relationships between the subsets of relations. The type of knowledge in this schema consists of the inherited facts specifying how the subsets of relations are interrelated to each other. Axioms of this type mostly carry how the relation could be subdivided and how the subrelations are interrelated (e.g., (6.4)).

<u>IA schema</u> : Let an n-ary relation R_1 and an m-ary relation R_2, whose attribute index sets are $\{ 1, \cdots, n \}$ and $\{ 1, \cdots, m \}$ respectively, be related by an axiom of RA on the attributes $\overline{X}_A$, where $A \subseteq \{ 1, \cdots, n \}$. Then a relationship between some fractions of these two relations R_1 and R_2 is expressed in the form of $\forall \overline{X}_A \ (\exists \overline{X}_{A'} \ \exists \overline{X}_{A^*} \, \psi(\overline{X}_A, \overline{X}_{A'}, \overline{X}_{A^*}) \rightarrow \exists \overline{Y}_B \ R_2(\overline{X}_A, \overline{Y}_B))$ where some variables of $\overline{X}_{A'}$ and at least one of each of $\overline{X}_{A^*}$ and $\overline{Y}_B$ are *aggregate variables* ; $A' = \{1, \cdots, n\} - A$, $B = \{1, \cdots, m\} - A$, A^* is some attribute index set of relations RA-related with R_1 ; and $\psi(\overline{X}_A, \overline{X}_{A'}, \overline{X}_{A^*}) \in Form(L_\Sigma)$ is a conjunction of atomic formulas of L_Σ including $R_1(\overline{X}_A, \overline{X}_{A'})$.

Here it is noticed that any axiom in this schema is a Σ-Horn formula (Σ-*Horn formula*† is a variation of Horn formula [Horn51] in which variables in the formula may be aggregate variables).

---

† In a more formal way, Σ-Horn formula can be stated as follows: A $\psi \in Form(L_\Sigma)$ is a *basic Σ-Horn formula* iff ψ is a disjunction of formulas Θ_i , $\psi = \Theta_1 \cup \cdots \cup \Theta_m$, where at most one of the formulas Θ_i is an atomic formula, and the rest of them are the negations of atomic formulas. A *Σ-Horn formula* is built up from the basic Σ-Horn formula with the connective $\cap$, the quantifiers $\exists$ and $\forall$. A *Σ-Horn sentence* is a Σ-Horn formula with no free variables.

(iv) Ground Defining Axiom (GDA)

In the axiom schemas of IA, some unary predicates are introduced as the accompanying symbol of aggregate variables. As previously discussed in Section 3.3, whenever a unary predicate, say P , is introduced syntactically, the meaning of the symbol P must also be described and introduced as what is call a defining axiom. There are two ways of doing this: one by GDA schema and the other by VDA schema. If the defining axiom is explicitly stated in terms of constants, it is called an axiom of this GDA schema and if it is implicitly defined in terms of some other existing relation predicates, then it is called an axiom of VDA schema. By the GDA schema, the members of the set which is interpreted by the introduced unary predicate are explicitly defined in terms of the constants of L_Σ. The GDA schema is formally stated as follows (e.g., (6.1)).

GDA schema : Given a unary predicate P to be introduced, the GDA schema is represented in the form, $\forall x \; (P(x) \leftrightarrows (x = c^1 \cup \cdots \cup x = c^n))$, where c^i , $1 \leq i \leq n$, is a constant symbol of a sort domain to which the aggregate variable accompanying P belongs.

(v) Virtual Defining Axiom (VDA)

An axiom of VDA schema is an axiom defining a unary predicate in terms of some other pre-existing predicates. When defining the set designed by a unary predicate, it may not only be described in terms of the constants of L_Σ but also may be expressed in terms of nonunary predicates of L_Σ . The set designated by a unary predicate may simply be expressed in terms of VDA schema by some combination of join, selection, and projection of relation predicates, instead of by only the constants of L_Σ. The formal representation schema is the following (e.g., (6.8)).

VDA schema : Given a unary predicate P to be introduced, the VDA schema is represented in the form, $\forall x\ (P(x) \leftrightarrows \alpha(x))$, where $\alpha(x) \in Form(L_\Sigma)$ and x is the only free variable in $\alpha(x)$.

6.2. Σ-Horn Knowledge base

In the previous section, five types of axiom schemas have been identified whose axioms would constitute the knowledge base of the KBS. The knowledge that is directly applied to the queries is in fact a special class of the axioms of the five schemas. Other kinds of knowledge are used for secondary purposes. In this section, it is discussed how the special class of the knowledge is constructed from the axioms of the five types of schemas.

The knowledge base of KBS can be said to consist of two levels that are denoted by KB and $KB_{\Sigma H}$, respectively. KB is the knowledge base constituting the five types of axioms that was introduced in the previous section and $KB_{\Sigma H}$ is a subset of the logical consequences of KB. In fact, KB is a proper subset of the complete theory $KB(DB)$ defined on the database structure DB, i.e., $KB \subset KB(DB)$. By this it is meant that KB is a collection of knowledge selected from $KB(DB)$ that is of interest to the DB designer of a specific application domain. In our case, the knowledge needed to estimate the URC's is the content of KB.

$KB_{\Sigma H}$ is indeed the collection of the axioms that actually take part in the syntactic inference procedure of the KBS (how it is done is the content of Section 7.2). $KB_{\Sigma H}$ consists of two types of axioms: [type I] the IA axioms in KB each of which has its "corresponding" FDA axiom in KB (the notion of "corresponding" is defined shortly), and [type II] a class of axioms equivalent to the IA axioms of type I which are individually derived from some relevant IA, FDA, GDA, VDA and RA axioms. The axioms of type II are also IA axioms. From the fact that both types of the axioms in $KB_{\Sigma H}$ are IA axioms and the IA axioms are Σ-Horn formulas, $KB_{\Sigma H}$ is called a Σ-Horn knowledge base. In the following it is shown by example how axioms of the five schemas constitute KB and how $KB_{\Sigma H}$ is constructed from KB . In this example it is also clarified why the five types of axioms are identified as useful knowledge for the KBS.

First, it is illustrated that the axioms of three schema types, FDA, IA and GDA, constitute KB . Let a knowledge provided by a DB designer be "All the dealers which are supplied car items are the car dealers." Let $B47$, $V01$, $V03$, and $W09$ stand for all the car items, and let 50 and 51 stand for all the car dealer types. If P and Q are defined, respectively,

$$\begin{aligned} &\forall x \ (P(x) \leftrightarrows (x=B47 \cup x=V01 \cup x=V03 \cup x=W09)) , \\ &\forall x \ (Q(x) \leftrightarrows (x=50 \cup x=51)) , \end{aligned} \qquad (6.1)$$

then, with a little exercise of imagination, the preceding knowledge is recapitulated by the following formulas in an ordinary many-sorted language:

$$\forall y \ (\exists x \ \exists z (Sa(x,y,z) \cap P(z)) \rightarrow \exists u \ \exists v \ (De(y,u,v) \ \cap \ Q(v))) \qquad (6.2)$$

in conjunction with a functional dependency in the relation *DEALERS* from *d#* to *d_type* of the following form

$$\forall x \ \forall y \ \forall z \ \forall y' \ \forall z' (De(x,y,z) \cap De(x,y',z') \rightarrow z=z') \qquad (6.3)$$

(why (6.2) must be in conjunction with (6.3) is additionally explained in detail later in Section 7.2). By introducing the aggregate variables $z^{\Sigma P}$ and $v^{\Sigma Q}$, (6.2) is equivalently expressed as

$$\forall y \ (\exists x \ \exists z^{\Sigma P} \ Sa(x,y,z^{\Sigma P}) \rightarrow \exists u \ \exists v^{\Sigma Q} \ De(y,u,v^{\Sigma Q})) . \tag{6.4}$$

Here (6.3) is a FDA axiom and (6.4) is an IA axiom. (6.3) and (6.4) are elements of KB . In the process of generating (6.4), it is required to add the axioms of (6.1) in KB as GDA axioms in order to make P and Q meaningful predicate symbols. The preceding illustration shows the axioms of the three schemas, FDA, IA and GDA, are elements of KB . Axioms of other two schemas, RA and VDA, are illustrated in the context of constructing $KB_{\Sigma H}$ from KB .

In the rest of the section, it is shown how $KB_{\Sigma H}$ is constructed from KB . First, the notion of "corresponding" FDA axiom is introduced for each IA axiom. Let an IA axiom K be of the form

$$K = \forall x_1 \cdots \forall x_n \ (\psi(x_1, \cdots, x_n) \rightarrow \exists z_1 \cdots \exists z_k \ R(x_{i_1}, \cdots, x_{i_h}, z_1, \cdots, z_k)),$$

where $\{x_{i_1}, \cdots, x_{i_h}\} \subseteq \{x_1, \cdots, x_n\}$. Let $z_{v_1}, \cdots, z_{v_l}$ be the only aggregate variables among $z_1, \cdots, z_k$. A FDA axiom is said to be ***the corresponding axiom of*** K if the FDA axiom is of the form

$$\forall u_1 \cdots \forall u_h \forall y_1 \cdots \forall y_k \forall y_1' \cdots \forall y_k' \ (R(u_1, \cdots, u_h, y_1, \cdots, y_k) \cap$$
$$R(u_1, \cdots, u_h, y_1', \cdots, y_k') \rightarrow y_{v_1} = y_{v_1}' \cap \cdots \cap y_{v_l} = y_{v_l}'),$$

which states that there is a functional dependency in R from $u_1, \cdots, u_h$ to $y_{v_1}, \cdots, y_{v_l}$.

$KB_{\Sigma H}$ is constructed by including only the IA axioms in KB each of which has its corresponding FDA axiom in KB . The IA axiom of (6.4) is considered. It is clear that (6.3) is the corresponding FDA axiom of the IA axiom (6.4). Since (6.4) is accompanied by (6.3) in KB , (6.4) is a legitimate element of $KB_{\Sigma H}$. There can be possibly a class of IA axioms in KB which are not accompanied by their corresponding FDA axioms. For instance, (6.4) may not be accompanied by (6.3) in KB , although (6.4) is still an IA axiom and is therefore an element of KB . Such IA axiom must not be included in $KB_{\Sigma H}$. The reason that IA axioms without their corresponding FDA axioms should not be included in $KB_{\Sigma H}$ is because they can lead to an incorrect identification of the *URC*'s (it will be discussed in detail later in Section 7.2).

Now it is illustrated how a class of axioms equivalent to the IA axioms in $KB_{\Sigma H}$ is derived and also included in $KB_{\Sigma H}$. In this context, the need of RA and VDA schemas is illustrated. The reason $KB_{\Sigma H}$ is expanded by adding new IA axioms is to allow a larger class of user queries to be handled by the KBS. These new axioms are also IA axioms and they are generated in conjunction with some relevant IA, FDA, GDA, VDA and RA axioms.

First it is shown how a VDA axiom is derived from IA, FDA and GDA axioms. Let there be a relationship saying "All the values of *i_type* in *ITEMS* for the car items are only 'bus', 'sedan' and 'van' " which indeed holds in the relation *ITEMS* [cf. Figure 6.1]. This relationship is expressed as

$$\forall x^{\Sigma P} \; (\exists y \; \exists z \; It(x^{\Sigma P}, y, z) \rightarrow \exists u \; \exists v^{\Sigma R} \; It(x^{\Sigma P}, u, v^{\Sigma R})) , \qquad (6.5)$$

where R is defined

$$\forall x \; (R(x) \leftrightarrows (x = sedan \cup x = bus \cup x = van)) , \qquad (6.6)$$

in conjunction with the functional dependency in *ITEMS* from *item#* to *i_type*

ITEMS

item#	name	i_type
C06	white 7	paint
N11	squ. 11"	nut
P02	distribu.	engin
P03	radiator	engin
S01	In. 8080	elect.
S02	battery	elect.
X89	iron 9"	plate
B47	Eland	bus
V01	Astre	sedan
V03	Camaro	sedan
W09	Brat	van

The following holds:

$$DEF(DB\ , It(z^{\Sigma P}, w, t)) .$$

$$=$$

$$DEF(DB\ , It(z, w, t^{\Sigma R})) .$$

Figure 6.1. Derivation of a VDA axiom

$$\forall x\ \forall y\ \forall z\ \forall y'\ \forall z' (It(x, y, z) \cap It(x, y', z') \rightarrow z = z') . \tag{6.7}$$

Here (6.5) is an IA axiom, (6.6), a GDA axiom and (6.7), a FDA axiom. The IA axiom (6.5), the FDA axiom (6.7) and two GDA axioms, one for R in (6.6) and the other for P in (6.1), imply that the unary predicate P which was once defined in terms of constants can now be defined in terms of the predicate It . That is, from (6.1), (6.5), (6.6) and (6.7), it follows that $P^{DB} = DEF(DB, \exists w\ \exists t^{\Sigma R}\ It(z, w, t^{\Sigma R}))$ [cf. Figure 6.1]. The meaning of P can now be expressed as

$$\forall z\ (P(z) \leftrightarrows\ \exists w\ \exists t^{\Sigma R}\ It(z, w, t^{\Sigma R})) . \tag{6.8}$$

(6.8) is a VDA axiom that is therefore an element of KB .

Now it is illustrated how an IA axiom in $KB_{\Sigma H}$ in conjunction with a VDA axiom and a RA axiom leads to another IA axiom that is equivalent to the IA axiom.

The IA axiom (6.4) and the VDA axiom (6.8) are considered. If the aggregate variable $z^{\Sigma P}$ shown in the antecedent of the IA axiom (6.4) is unraveled, (6.4) is equivalently expressed as

$$\forall y(\exists x\ \exists z(Sa(x,y,z)\cap P(z))\rightarrow \exists u\ \exists v^{\Sigma Q} De(y,u,v^{\Sigma Q}))\,. \tag{6.9}$$

Then (6.9) and the VDA axiom (6.8) suggest a way to provide an axiom that is equivalent to (6.4). That is, $P(z)$ in (6.9) may simply be replaced by $\exists w\ \exists t^{\Sigma R}\ It(z,w,t^{\Sigma R})$ without changing its meaning as long as the equivalence of these two expressions are defined in terms of the VDA axiom (6.8). However, the replacement of such unary predicate by using a VDA axiom should not be made unless there is a RA axiom in KB which is called the "relevant" axiom to doing so.

The notion of a "relevant" RA axiom is introduced as follows: Let K be an IA axiom describing a relationship between some subsets of two relations, say R_1 and R_2, of the form

$$K = \forall x_1 \cdots \forall x_n\ (\psi(x_{i_1}, \cdots, x_{i_m})\cap R_1(x_{j_1}, \cdots, x_{j_l})\rightarrow$$
$$\exists z_1 \cdots \exists z_k\ R_2(x_{a_1}, \cdots, x_{a_k}, z_1, \cdots, z_k))$$

where $\{x_{a_1}, \cdots, x_{a_k}\}\subseteq\{x_1, \cdots, x_n\}$ and $\{x_{i_1}, \cdots, x_{i_m}, x_{j_1}, \cdots, x_{j_l}\} = \{x_1, \cdots, x_n\}$. Let x_{j_r} be an aggregate variable whose range is restricted by a unary predicate, say P, where $x_{j_r}\in\{x_{j_1}, \cdots, x_{j_l}\}$ and $x_{j_r}\notin\{x_{a_1}, \cdots, x_{a_k}\}$, and let this aggregate variable be unraveled as follows:

$$K' = \forall x_1 \cdots \forall x_n\ (\psi(x_{i_1}, \cdots, x_{i_m})\cap R_1(x_{j_1}, \cdots, x_{j_l})\cap P(x_{j_r})\rightarrow$$
$$\exists z_1 \cdots \exists z_k\ R_2(x_{a_1}, \cdots, x_{a_k}, z_1, \cdots, z_k))$$

where x_{j_r} is now no longer an aggregate variable. Let there be a VDA axiom of the form,

$$\forall x_{j_r}\ (P(x_{j_r}) \leftrightarrows \exists y_{i_1} \cdots \exists y_{i_v} R_3(y_1, \cdots, y_{v+1}))$$

where $\{x_{j_r}, y_{i_1}, \cdots, y_{i_v}\} = \{y_1, \cdots, y_{v+1}\}$. Then it is said that a RA axiom of the form

$$\exists u_{j_1} \cdots \exists u_{j_l} \exists w_1 \cdots \exists w_{v+1} (R_1(u_{j_1}, \cdots, u_{j_l}) \cap R_3(w_1, \cdots, w_{v+1}) \cap (u_{j_r} = w_{g_r}))$$

is *the relevant RA axiom to the replacement of* $P(x_{j_r})$ *in* K' *by* $\exists y_{i_1} \cdots \exists y_{i_v} R_3(y_1, \cdots, y_{v+1})$, where $u_{j_r} \in \{u_{j_1}, \cdots, u_{j_l}\}$, $w_{g_r} \in \{w_1, \cdots, w_{v+1}\}$ and w_{g_r} is the $x_{j_r} \in \{y_1, \cdots, y_{v+1}\}$.

When replacing $P(x_{j_r})$ in K' by $\exists y_{i_1} \cdots \exists y_{i_v} R_3(y_1, \cdots, y_{v+1})$, the relevant RA axiom is needed because the presence of the relevant RA axiom implies the resulting formula obtained by the replacement would be useful. By definition, the RA axiom of the preceding form means join of R_1 and R_3 over the attribute indicated by x_{j_r} is meaningful. This implies that queries including the join of R_1 and R_3 over the attribute indicated by x_{j_r} are meaningful, which therefore means the resulting formula obtained by the replacement can be used to restrict such queries.

The use of relevant RA axiom is illustrated in the following. Let the join of the two relations *SALES* and *ITEMS* via *item#* be meaningful in the sense that queries including the join of the two relations on *item#* are meaningful. This relationship is expressed as

$$\exists x \exists y \exists z \exists u \exists w \exists t\ (\ Sa(x,y,z) \cap It(u,w,t) \cap z=u\ ). \qquad (6.10)$$

Here (6.10) is a RA axiom which is therefore an element of *KB*. Furthermore (6.10) is the relevant RA axiom to the replacement of $P(z)$ in (6.9) by $\exists w\ \exists t^{\Sigma R}\ It(z,w,t^{\Sigma R}))$ in (6.8). Replacing $P(z)$ by $\exists w\ \exists t^{\Sigma R}\ It(z,w,t^{\Sigma R}))$ rephrases (6.9) into the following form:

$$\forall y\ (\exists x\ \exists z\ \exists w\ \exists t^{\Sigma R}(Sa(x,y,z)\ \cap\ It(z,w,t^{\Sigma R}))\rightarrow\ \exists u\ \exists v^{\Sigma Q}\ De(y,u,v^{\Sigma Q})). \quad (6.11)$$

(6.9) describes the same knowledge described by (6.4) in a different way "All the dealers who are supplied 'sedan', 'bus' and 'van' are the car dealers". Here, (6.11) is again an IA axiom that was intended to be derived. (6.11) is an element of $KB_{\Sigma H}$. At the end of Section 7.1, it will be illustrated how the inclusion of the new IA axioms such as (6.11) enlarges the class of queries to be handled by the KBS. It is noticed that although $KB_{\Sigma H}$ contains only IA axioms (which are all Σ-Horn formulas), the other types of axioms have been indirectly embedded in the construction of $KB_{\Sigma H}$.

CHAPTER VII

INFERENCE PROCEDURE

7.1. Inference Procedure

In this section, it is described how the knowledge about the data, i.e., $KB_{\Sigma H}$, is applied to the queries, i.e., query clauses in Σ-normal form, in order to lead to an equivalent query clause which shows more precise *URC*'s than does the original query. This process is done by the inference procedure of the KBS. Let the inference procedure be abbreviated by the symbol "$\longmapsto$". Then "$\longmapsto$" requires some preliminary steps to be made for the formulas in $KB_{\Sigma H}$ and the query expressions in Σ-normal form. Each formula in $KB_{\Sigma H}$ is converted into an existential quantifier free form by the process known as Skolemization. Once the Skolemization step is performed, all the quantifiers can be omitted from the formulas in $KB_{\Sigma H}$ and the query expressions. This is possible because the formulas in $KB_{\Sigma H}$ are only universally quantified and the query expressions are only existentially quantified.

The step converting each formula in $KB_{\Sigma H}$ into an existential quantifier free form is described in detail. First, the formulas in $KB_{\Sigma H}$ are converted into the logically equivalent prenex normal forms. Then the prenex normal forms are converted into existential quantifier free forms by the usual procedure called Skolemization. Here the Skolemization for the formulas of L_Σ differs from the ordinary Skolemization only by the fact that when a Skolem function is introduced, its range must be restricted to a unary relation which is the same as the range of the variable to be replaced by the function. An example illustrates the Skolemization process for a formula in L_Σ:

Example 7.1.1

Consider the IA axiom of (6.4). Let this axiom be ψ,

$$\psi = \forall y \ (\exists x \ \exists z^{\Sigma P} \ Sa(x, y, z^{\Sigma P}) \rightarrow \exists u \ \exists v^{\Sigma Q} \ De(y, u, v^{\Sigma Q})) .$$

The logically equivalent prenex normal form of ψ is

$$\forall y \ \forall x \ \forall z^{\Sigma P} \ \exists u \ \exists v^{\Sigma Q} \ (Sa(x, y, z^{\Sigma P}) \rightarrow De(y, u, v^{\Sigma Q})) .$$

Its Skolemized form is

$$\forall y \ \forall x \ \forall z^{\Sigma P} \ (Sa(x, y, z^{\Sigma P}) \rightarrow De(y, g(y, x, z^{\Sigma P}), f^Q(y, x, z^{\Sigma P}))) ,$$

where $g(y, x, z^{\Sigma P})$ and $f^Q(y, x, z^{\Sigma P})$ are the Skolem functions replaced for the variables u and $v^{\Sigma Q}$, respectively. It is noticed that the range of $f^Q(y, x, z^{\Sigma P})$ is denoted by the superscript Q which is the range of $v^{\Sigma Q}$.

The Skolemization step needs no justification as long as the Skolemized formulas are equivalent to the formulas prior to Skolemization. Once the Skolemization step is completed, " $\longmapsto$" manipulates only the matrices of the Skolemized formulas in $KB_{\Sigma H}$ with the matrices of the existentially closed query expressions. As stated previously, as long as all the formulas in $KB_{\Sigma H}$ are only universally quantified and all the query expressions are only existentially quantified, the presence of the quantifiers can be made implicit in the symbol manipulation process " $\longmapsto$ ".

In order to describe " $\longmapsto$ ", two notions, namely, "match" and "restrictable", need to be defined. Before defining these notions, a few notations are first introduced in the following: After all the formulas of $KB_{\Sigma H}$ are Skolemized and their universal quantifiers are stripped off, let the resulting set of matrices be denoted $KB^{m}_{\Sigma H}$. After the existential quantifiers are stripped off from all the queries concerned, let the resulting set of the query clauses be denoted by Q_c^m . For $q \in Q_c^m$, let $SUB(q)$ stand for the collection of all the subformulas of q . Each formula in $SUB(q)$ is again a conjunction of atomic formulas. Since any formula in $KB_{\Sigma H}$ is of IA schema, it follows that any formula in $KB^{m}_{\Sigma H}$ is of the form $\psi \rightarrow R(t_1, \cdots, t_n)$ where ψ is a conjunction of atomic formulas and $R(t_1, \cdots, t_n)$ is an atomic formula with R being a relation predicate and some of the terms among $t_1, \cdots, t_n$ being Skolem functions [IA schema is defined in such a way that there is at least one existentially quantified variable in the consequent. See (ii) of Section 6.1]. This formality is used in defining the two notions, "match" and "restrictable".

The notions of "match" and "restrictable" are the following. For $q \in Q_c^m$ and $K_j \in KB^{m}_{\Sigma H}$ with K_j of the form $\psi_j \rightarrow R(t_1, \cdots, t_n)$, some $q_i \in SUB(q)$ that does not include the predicate R in it *matches* (or "*is matched by*") K_j if the two following conditions are satisfied:

(1) A predicate symbol is in q_i if and only if the same predicate symbol is in ψ_j .

(2) $DEF(DB, q_i) \subseteq DEF(DB, \psi_j)$.

A query clause $q \in Q_c^m$ is *restrictable* by an element $K_j \in KB^{m}_{\Sigma H}$ if the following two conditions are satisfied:

(1) There is $q_i \in SUB(q)$ which matches K_j .

(2) For the consequent $R(t_1, \cdots, t_n)$ of K_j , there is an atomic formula $R_q(t_1^q, \cdots, t_n^q)$ in q such that (i) R and R_q are identical relation predicates, and (ii) there is a variable $t_l^q \in \{t_1^q, \cdots, t_n^q\}$ and a Skolem function $t_l \in \{t_1, \cdots, t_n\}$ such that $Ran(t_l) \subset Ran(t_l^q)$, where by $Ran(t)$ is meant the range of the outermost symbol of the term t .

Here (R, R_q) in (2) is called ***a restriction pair associated with q_i and K_j*** in (1). If q is restrictable by K_j , it is said that q ***is restricted by K_j*** using the following process: for each restriction pair (R, R_q), if the variable t_l^q in R_q and the Skolem function t_l in R satisfy the relationship $Ran(t_l) \subset Ran(t_l^q)$, then substitute t_l^q by a variable v whose range $Ran(v) = Ran(t_l)$. Here t_l^q in R_q is called ***the corresponding variable of t_l in R*** . The restricted q is denoted by $q \mid K_j$.

Now the inference procedure "$\longmapsto$" is introduced in the following:

Inference Procedure "$\longmapsto$"

Step 1 Let $q^* = q$, $W = SUB(q)$, and go to Step 2.

Step 2 Let q_i be an element of W, and go to Step 3.

Step 3 Let $MATCH(q_i)$ be all the formulas in $KB_{\Sigma H}^m$ which match q_i . If $MATCH(q_i)$ is empty, go to Step 5 ; otherwise go to Step 4.

Step 4 Do while $MATCH(q_i)$ is not empty,

1. let K_j be an element of $MATCH(q_i)$,
2. $MATCH(q_i) = MATCH(q_i) - K_j$, and

3. let $q^* = q^* \mid K_j$ only if q^* is restrictable by K_j ;

and go to Step 5.

Step 5 Let $W = W - q_i$. If W is empty, stop ; otherwise, go to Step 2.

The preceding procedure always stops at Step 5 either (i) with q^* being a revised version of $q \in Q_c^m$ if there was any element in $KB_{\Sigma H}^m$ which restricted q , or (ii) with q^* being identical with q otherwise. The complexity of the preceding procedure is discussed in the following. The following notations are used:

n the size of $KB_{\Sigma H}^m$.

r the number of atomic formulas in q .

$\delta(i)$ the number of atomic formulas in the i^{th} subformula q_i of q .

$\lambda(i)$ the number of free variables in the i^{th} subformula q_i of q .

A_k^i the size of the range of the k^{th} free variable, $1 \leq k \leq \lambda(i)$, in the i^{th} subformula q_i of q .

$\eta(j)$ the number of free variables in the antecedent of the j^{th} knowledge K_j in $KB_{\Sigma H}^m$.

B_l^j the size of the range of the l^{th} free variable, $1 \leq l \leq \eta(j)$, in the antecedent of the j^{th} knowledge K_j in $KB_{\Sigma H}^m$.

$\xi(j)$ the number of the Skolem functions in the consequent of the j^{th} knowledge K_j in $KB_{\Sigma H}^m$.

C_m^j the size of the range of the m^{th} Skolem function, $1 \leq m \leq \xi(j)$, in the consequent of the j^{th} knowledge K_j in $KB_{\Sigma H}^m$.

D_m^j the size of the range of the corresponding variable in R_q^j of the m^{th} Skolem function in the consequent of the j^{th} knowledge K_j in $KB_{\Sigma H}^m$ where R_q^j is the atomic formula with which the consequent of K_j constitutes a restriction pair.

At Step 1, the total number of possible subformulas of q is $2^r - 1$, i.e., for the set W of subformulas of q its size $|W| = 2^r - 1$. This means that the outermost loop (i.e., Step 2 $\rightarrow$ Step 5 $\rightarrow$ Step 2) of the procedure is repeatedly carried out as many times as $2^r - 1$ at most. At Step 3, finding $MATCH(q_i)$ entails comparing each member of $KB_{\Sigma H}^m$ with q_i. This comparison consists of two types of testings. First, for each element, say the j^{th} member K_j of $KB_{\Sigma H}^m$, it needs to determine whether all the predicate symbols of q_i are in the antecedent ψ_j of K_j and no other predicate is in ψ_j. Since both q_i and ψ_j do not contain any predicate symbol more than once in their expressions, the worst case of determining the preceding condition is when q_i and ψ_j both have $\delta(i)$ atomic formulas. Determining the preceding condition requires comparisons of no more than

$$\delta(i)^2 .$$

Second, for each knowledge satisfying the preceding condition, say K_j, it needs to determine between q_i and the antecedent ψ_j of K_j whether $DEF(DB(Auto), q_i) \subseteq DEF(DB(Auto), \psi_j)$. Since $|DEF(DB(Auto), q_i)| \leq \prod_{k=1}^{\lambda(i)} A_k^i$ and $|DEF(DB(Auto), \psi_j)| \leq \prod_{l=1}^{\eta(j)} B_l^j$, this determination requires comparisons of no more than

$$\prod_{k=1}^{\lambda(i)} A_k^i (\log \prod_{k=1}^{\lambda(i)} A_k^i) + \prod_{k=1}^{\lambda(i)} B_k^j (\log \prod_{k=1}^{\lambda(i)} B_k^j) + Max(\prod_{k=1}^{\lambda(i)} A_k^i \ , \ \prod_{k=1}^{\lambda(i)} B_k^j)$$

[notice that both q_i and ψ_j of K_j have $\lambda(i)(=\eta(j))$ free variables]. Let the preceding term be abbreviated by $P(i\ ,\ j)$. Since q_i is compared with each individual in $KB_{\Sigma H}$, the overall complexity of generating $MATCH(q_i)$ at Step 3 requires comparisons of at most

$$\sum_{j=1}^{n} (\delta(i)^2 + P(i\ ,\ j)) .$$

At Step 4, the Do-while loop is processed as many times as $|MATCH(q_i)|$. The restriction step, i.e., (3) in the Do-while loop, requires to test $Ran(t_m) \subset Ran(t_m^q)$ where t_m is the m^{th} Skolem function, $1 \leq m \leq \xi(j)$, in the consequent of K_j and t_m^q is its corresponding variable in q. This test requires comparisons of at most

$$\sum_{m=1}^{\xi(j)} C_m^j D_m^j .$$

Since the Do-while loop is processed as many times as $|MATCH(q_i)|$, and $|MATCH(q_i)|$ is at most n [$|MATCH(q_i)| = n$ is the case when all the members of $KB_{\Sigma H}^m$ match q_i], the total number of comparisons at Step 4 is no more than

$$\sum_{j=1}^{n} \sum_{m=1}^{\xi(j)} C_m^j D_m^j .$$

Hence overall the total number of comparisons to be made in the procedure "$\longmapsto$" is no more than

$$\sum_{i=1}^{2^r-1} [\sum_{j=1}^{n} (\delta(i)^2 + P(i\ ,\ j)) + \sum_{j=1}^{n} \sum_{m=1}^{\xi(j)} C_m^j D_m^j] .$$

Let L stand for the number of atomic formulas in the longest possible query in

Q_c^m . Then for any i , $1 \le i \le 2^r - 1$, $\delta(i) \le L$. Let M stand for the size of the largest sort domain of the given database application structure DB . Then for any i , $1 \le i \le 2^r - 1$, and k , $1 \le k \le \lambda(i)$, $A_k^i \le M$ and for any j , $1 \le j \le n$, and l , $1 \le l \le \eta(j)$, $B_l^j \le M$. Also for any j , $1 \le j \le n$, and m , $1 \le m \le \xi(j)$, $C_m^j \le M$ and $D_m^j \le M$. Let K be the largest possible value for $\lambda(i)$ and $\xi(j)$ where $1 \le i \le 2^r - 1$ and $1 \le j \le n$. The following relationship holds:

$$P(i\ ,j) = \prod_{k=1}^{\lambda(i)} A_k^i(\log \prod_{k=1}^{\lambda(i)} A_k^i) + \prod_{k=1}^{\lambda(i)} B_k^j(\log \prod_{k=1}^{\lambda(i)} B_k^j) + Max(\prod_{k=1}^{\lambda(i)} A_k^i\ ,\ \prod_{k=1}^{\lambda(i)} B_k^j)$$
$$\le M^{\lambda(i)}\log M^{\lambda(i)} + M^{\lambda(i)}\log M^{\lambda(i)} + M^{\lambda(i)} .$$

Using O-notation the overall complexity of " $\longmapsto$ " is concluded as follows:

$$\sum_{i=1}^{2^r-1} [\ \sum_{j=1}^{n} (\delta(i)^2 + P(i\ ,j)) + \sum_{j=1}^{n}\sum_{m=1}^{\xi(j)} C_m^j D_m^j\ ]$$
$$\le \sum_{i=1}^{2^r-1}\sum_{j=1}^{n} [\ L^2 + (\ M^{\lambda(i)}\log M^{\lambda(i)} + M^{\lambda(i)}\log M^{\lambda(i)} + M^{\lambda(i)}\ ) + M^2 \sum_{m=1}^{\xi(j)} 1\ ]$$
$$\le \sum_{i=1}^{2^r-1}\sum_{j=1}^{n} [\ L^2 + M^K(\ 2\log M^K\ + 1\ ) + M^2K\ ]$$
$$\le [\ L^2 + M^K(\ 2\log M^K\ + 1\ ) + M^2K\ ]O(2^r)O(n)$$
$$= M'O(2^r)O(n) \quad \text{for some constant } M' .$$

Although the overall complexity includes the exponentially growing term $O(2^r)$, it is expected that $O(2^r)$ is limited to a certain constant value since in most cases a query does not involve more than 3 or 4 atomic formulas. Therefore, if the $O(2^r)$ term is replaced by some constant, say C , and if $C' = CM'$ for some constant C' , then the overall complexity of $\longmapsto$ is

$$C' O(n)$$

where n is the size of the knowledge base $KB^m_{\Sigma H}$.

The procedure " $\longmapsto$" produces two results: if " $\longmapsto$" applies $KB^m_{\Sigma H}$ to $q \in Q^m_c$ to produce q^* [from here on, the whole procedure will be abbreviated by $KB^m_{\Sigma H}$; $q \longmapsto q^*$], then (i) q^* is "equivalent" to q and (ii) if $q^* \neq q$ then q^* shows "more precise" URC's than q . By the equivalence between q^* and q it is meant that $DEF(DB, q) = DEF(DB, q^*)$. Let q and q^* be $R_1 \cap \cdots \cap R_m$ and $R_1^* \cap \cdots \cap R_m^*$, respectively, where R_i and R_i^*, $1 \leq i \leq m$, are atomic formulas of L_Σ. Then according to Definition 5.2.3, the URC's identified by q and q^* are $\bigcup_{i \in \{1, \cdots, m\}} DEF^i(DB, q)$ and $\bigcup_{i \in \{1, \cdots, m\}} DEF^i(DB, q^*)$, respectively, where for each i $DEF^i(DB, q) = \{DEF(DB, R_i)\}$ and $DEF^i(DB, q^*) = \{DEF(DB, R_i^*)\}$. By the fact that q^* shows more precise URC's than q it is meant the following relationships hold: (i) for no j , $1 \leq j \leq m$, $DEF(DB, R_j) \not\subset DEF(DB, R_j^*)$, and (ii) for some i , $1 \leq i \leq m$, $DEF(DB, R_i^*) \subset DEF(DB, R_i)$.

Showing the equivalence of q^* and q in a formal way is the content of Section 7.2. In the following it is first demonstrated that q^* shows more precise URC's than q along with illustrating " $\longmapsto$" by an example.

<u>Example 7.1.2</u>

Let the query q_1 in Example 5.2.2 be concerned with,

$$q_1 = \exists x \; \exists y \; \exists z^{\Sigma I} \; \exists v \; (Sa(x, y, z^{\Sigma I}) \cap De(y, u, v)) . \tag{7.1}$$

Here it is shown how the IA axiom (6.4) in $KB_{\Sigma H}$ is applied to q_1 of (7.1) to derive q^* in a purely syntactic way by "$\mapsto$". In Example 7.1.1, it has been shown that the IA axiom (6.4) can be Skolemized into the following form:

$$\forall y \ \forall x \ \forall z^{\Sigma P} \ (Sa(x,y,z^{\Sigma P}) \rightarrow De(y,g(y,x,z^{\Sigma P}),f^Q(y,x,z^{\Sigma P}))) , \qquad (7.2)$$

where $g(y,x,z^{\Sigma P})$ and $f^Q(y,x,z^{\Sigma P})$ are Skolem functions. (7.2) clearly shows how the relations *SALES* and *DEALERS* are related fragment by fragment, namely, *CAR_SALES* of *SALES* and *CAR_DEALERS* of *DEALERS*. Now let the matrices of (7.1) and (7.2) be q and K_j, respectively. The followings hold: $Sa(x,y,z^{\Sigma I})$ in q matches K_j, since for the antecedent $Sa(x,y,z^{\Sigma P})$ of K_j it holds that (i) the predicate symbol Sa in q_i is the only predicate symbol in the antecedent of K_j, and (ii) $DEF(DB(Auto),Sa(x,y,z^{\Sigma I})) \subseteq DEF(DB(Auto), Sa(x,y,z^{\Sigma P}))$; and q is restrictable by K_j since there is a restriction pair $(De(y,g(y,x,z^{\Sigma P}),f^Q(y,x,z^{\Sigma P})), De(y,u,v))$ where $Ran(f^Q(y,x,z^{\Sigma P})) \subset Ran(v)$. Therefore, by substituting v in $De(y,u,v)$ by the variable $w^{\Sigma Q}$ whose range is identical to that of $f^Q(y,x,z^{\Sigma P}))$, q^* is concluded to be

$$q^* = Sa(x,y,z^{\Sigma I}) \cap De(y,u,w^{\Sigma Q}) . \qquad (7.3)$$

The *URC*'s indicated by q^* in (7.3) are the set of the defined relations $DEF(DB(Auto),Sa(x,y,z^{\Sigma I}))$ and $DEF(DB(Auto),De(y,u,w^{\Sigma Q}))$. When these are compared with the *URC*'s indicated by q, i.e., the set of the defined relations $DEF(DB(Auto),Sa(x,y,z^{\Sigma I}))$ and $DEF(DB(Auto),De(y,u,v))$, it is clear that q^* shows more precise *URC*'s than does q (see Figure 7.1).

The *URC*'s From *q*

$DEF(DB(Auto), Sa(x, y, z^{\Sigma P}))$

div#	d#	item#
01AP	01A	V01
02AP	01A	B47
04AP	01A	V03
05AP	55L	V03

and

$DEF(DB(Auto), De(y, u, v))$

d#	address	d_type
01A	Ann Arbor	51
03A	Dearborn	30
07A	Flint	50
26M	Cleveland	20
33B	Cleveland	30
48B	Rockford	31
55L	Flint	51
65B	Detroit	20
66L	Nile	23
70A	Lansing	70

The *URC*'s From *q'*

$DEF(DB(Auto), Sa(x, y, z^{\Sigma P}))$

div#	d#	item#
01AP	01A	V01
02AP	01A	B47
04AP	01A	V03
05AP	55L	V03

and

$DEF(DB(Auto), De(y, u, w^{\Sigma Q}))$

d#	address	d_type
01A	Ann Arbor	51
07A	Flint	50
55L	Flint	51

Figure 7.1. The *URC*'s Revealed to the Relations *SALES* and *DEALERS*

The result illustrated in the preceding example is formalized as follows:

Theorem 7.1.1

If $KB_{\Sigma H}^{m}$; $q \longmapsto q^{*}$ and $q^{*} \neq q$, then the following relationships hold between $\bigcup_{i \in \{1, \cdots, m\}} DEF^{i}(DB\ ,q)$ and $\bigcup_{i \in \{1, \cdots, m\}} DEF^{i}(DB\ ,q^{*})$: (i) for no j , $1 \leq j \leq m$, $DEF(DB\ ,R_j) \not\subset DEF(DB\ ,R_j^{*})$, and (ii) for some i , $1 \leq i \leq m$, $DEF(DB\ ,R_i^{*}) \subset DEF(DB\ ,R_i)$.

Proof. When q is restricted by a formula in $KB_{\Sigma H}^{m}$, only the following type of modifications is made on q : a variable, say x , in q is replaced by some variable, say w , satisfying the condition $Ran(w) \subset Ran(v)$. The theorem follows immediately. *Q.E.D.*

Having introduced the inference procedure "$\longmapsto$", it can be pointed out more clearly what advantages are obtained by using L_{Σ}^{1} as the tool for describing queries and the knowledge about the data. This can be discussed in the context of what problems could have occurred if the queries and the knowledge about the data were expressed in an ordinary many-sorted language (L_m).

When the queries and the knowledge about the data are expressed in L_m , symbolic manipulation of these two objects entails extra computation which is unnecessary if these two objects are expressed in L_{Σ}. Such extra computation is caused by the loss of "a form of meta knowledge" which otherwise is embedded and maintained *in the aggregate variables of* L_{Σ}. The query q_1 of (7.1) and the IA axiom, say K , of (6.4) are considered.

$$q_1 = \exists x\ \exists y\ \exists z^{\Sigma J}\ \exists v\ (Sa(x,y,z^{\Sigma J}) \cap De(y,u,v)).$$

$$K = \forall y\ (\exists x\ \exists z^{\Sigma P}\ Sa(x,y,z^{\Sigma P}) \rightarrow \exists u\ \exists v^{\Sigma Q}\ De(y,u,v^{\Sigma Q})).$$

Let the aggregate variables in the two formulas q_1 and K are unraveled into relativized expressions in L_m . Let q_1^m and K^m be the resulting relativized expressions equivalent to q_1 and K , respectively.

$$q_1^m = \exists x\ \exists y\ \exists z\ \exists v\ (Sa(x,y,z) \cap J^*(z) \cap De(y,u,v)),$$

$$K^m = \forall y\ (\exists x\ \exists z(Sa(x,y,z) \cap P^*(z)) \rightarrow \exists u\ \exists v\ (De(y,u,v) \cap Q^*(v))),$$

where the symbol $*$ is used to designate that the atomic formulas with $*$ are exclusively used for the purpose of variable range restriction. For convenience, from here on the following convention is made: although, strictly speaking, the range of a variable in the relativized expressions, such as z in q_1^m , is the sort to which the variable belongs, by the range of such a variable it will be meant the relation indicated by the atomic formula which has the variable as its only argument and is superscripted with $*$.

The two formulas q_1 and q_1^m are considered. In q_1 the range of $z^{\Sigma J}$ in $Sa(x,y,z^{\Sigma J})$ is determinable as the relation J from the variable itself since $z^{\Sigma J}$ itself contains the information about its own range. In contrast with this, in q_1^m the range of z in $Sa(x,y,z)$ can not be determined as the relation J unless *it is tested whether there is an atomic formula with* $*$ *in* q_1^m *which has* z *as its only argument and whose predicate symbol designates the relation* J , i.e., $J^*(z)$. When aggregate variables are unraveled into relativized expressions, such *range determination test* becomes necessary since unlike the aggregate variables the variables in the relativized expressions no longer contain the range restriction information on the variables themselves. Similar argument can be applied to $z^{\Sigma P}$ and $v^{\Sigma Q}$ of K and z and v of K^m .

What has been argued in the preceding paragraph is discussed in detail. It is shown why the range restriction test means extra computation in the symbolic manipulation. First it is formally stated how a formula in L_Σ is expressed in terms of relativized expression in L_m . Let a formula σ_Σ in L_Σ be

$$\sigma_\Sigma = \exists z^{\Sigma P} \_\_\_\_\_\_\_\_ z^{\Sigma P} \_\_\_\_\_\_\_\_ .$$

Then σ_Σ is translated into σ_m in L_m as follows:

$$\sigma_m = \exists z \; (\_\_\_\_\_\_\_\_ \; z \; \_\_\_\_\_\_\_\_ \cap P^*(z)) .$$

In σ_m the symbol $*$ is used as an aid to provide notational convenience, i.e., to indicate that the atomic formulas with $*$ are exclusively used for the purpose of variable range restriction.

Let the queries and the knowledge about the data which were expressed in L_Σ be expressed by the relativized expressions in L_m . Let an inference procedure, namely " $\models\!\!\rightarrow^m$ ", be developed which is applicable to the queries and the knowledge expressed in L_m . Let " $\models\!\!\rightarrow^m$ " consist of five steps each of whose function is identical with its corresponding step of " $\models\!\!\rightarrow$." Let the superscript m be used to indicate various notation used in " $\models\!\!\rightarrow^m$ " so that the notations used in " $\models\!\!\rightarrow^m$ " can be distinguished from its corresponding notations used in " $\models\!\!\rightarrow$," for instance, q^m and K_j^m are now the formulas in L_m . The inference procedure " $\models\!\!\rightarrow^m$ " is the following:

Inference Procedure "$\mapsto^m$"

Step 1^m Let $q^* = q^m$, $W^m = SUB^m(q^m)$, and go to Step 2^m.

Step 2^m Let q_i^m be an element of W^m, and go to Step 3^m.

Step 3^m Let $MATCH^m(q_i^m)$ be all the formulas in $KB_{\Sigma H}^{mm}$ which match q_i^m. If $MATCH^m(q_i^m)$ is empty, go to Step 5^m ; otherwise go to Step 4^m.

Step 4^m Do while $MATCH^m(q_i^m)$ is not empty,

1. let K_j^m be an element of $MATCH^m(q_i^m)$,
2. $MATCH^m(q_i^m) = MATCH^m(q_i^m) - K_j^m$, and
3. let $q^* = q^* \mid K_j^m$ only if q^* is restrictable by K_j^m;

and go to Step 5^m.

Step 5^m Let $W^m = W^m - q_i^m$. If W^m is empty, stop ; otherwise, go to Step 2^m.

"$\mapsto^m$" is different from "$\mapsto$" by the following: Let q_*^m be the subformula of q^m which does not include any atomic formulas with $*$. Then at Step 1^m $SUB^m(q^m)$ is constructed by including only all the subformulas of q_*^m. Doing so is appropriate since the atomic formulas with $*$ in q^m are irrelevant to constructing $MATCH^m(q_i^m)$ of Step 3^m. At Step 3^m, when $MATCH^m(q_i^m)$ is constructed the presence of the atomic formulas with $*$ is ignored in q_i^m and each member of $KB_{\Sigma H}^{mm}$. When determining whether all the predicate symbols of q_i^m are in the antecedent ψ_j^m of K_j^m and no other predicate is in ψ_j^m, atomic formulas with $*$ need not to be considered since their usage has nothing to do with determining what relations are involved in q_i^m and ψ_j^m.

The complexity of " $\longmapsto^m$ " is discussed. Let the following notations be additionally introduced:

ϵ the number of atomic formulas with $*$ in q^m .

$\alpha(i)$ the number of atomic formulas with $*$ in the i^{th} subformula q_i^m

$\beta(j)$ the number of atomic formulas with $*$ in the antecedent of the j^{th} knowledge K_j^m in $KB_{\Sigma H}^{mm}$.

$\gamma(j)$ the number of atomic formulas with $*$ in the consequent of the j^{th} knowledge K_j^m in $KB_{\Sigma H}^{mm}$.

At Step 1^m, since the total number of possible subformulas of q_i^m is also $2^r - 1$, $| W^m | = 2^r - 1$. This means that the outermost loop (i.e., Step 2^m $\rightarrow$ Step 5^m $\rightarrow$ Step 2^m) of the procedure " $\longmapsto^m$ " is also repeatedly carried out as many times as $2^r - 1$ at most.

At Step 3^m, since the presence of the atomic formulas with $*$ is ignored in constructing $MATCH^m(q_i^m)$, determining whether all the predicate symbols of q_i^m are in the antecedent ψ_j^m of K_j^m and no other predicate is in ψ_j^m requires the same complexity as that of " $\longmapsto$," i.e.,

$$\delta(i)^2 .$$

However, after the preceding condition has been tested, when determining whether $DEF(DB(Auto), q_i^m) \subseteq DEF(DB(Auto), \psi_j^m)$ additional comparisons are needed. Determining $DEF(DB(Auto), q_i) \subseteq DEF(DB(Auto), \psi_j)$ requires to know the ranges of the variables of q_i^m and ψ_j^m. Since the ranges of these variables are specified in terms of the atomic formulas with $*$, what has been called range determination test must be made for the variables in q_i^m and ψ_j^m, i.e., the ranges of the variables in q_i^m and ψ_j^m must be derived from the set of atomic formulas with $*$ in q_i^m and the set of atomic formulas with $*$ in ψ_j^m, respectively. Since the number of the atomic formulas with $*$ in q_i^m is $\alpha(i)$ and the number of variable in q_i^m is $\lambda(i)$, determining the ranges of the variables in q_i^m requires comparisons of no more than

$$\sum_{r=1}^{\lambda(i)} \alpha(i) = \alpha(i) \sum_{r=1}^{\lambda(i)} = \alpha(i)\lambda(i) .$$

Similarly, determining the ranges of the variables of ψ_j^m requires comparisons of no more than

$$\sum_{r=1}^{\eta(j)} \beta(j) = \beta(j) \sum_{r=1}^{\eta(j)} = \beta(j)\eta(j) .$$

Once the ranges of the variables in q_i^m and ψ_j^m are determined, the complexity of determining whether $DEF(DB(Auto), q_i) \subseteq DEF(DB(Auto), \psi_j)$ is identical with that of Step 3 of "$\longmapsto$." Thus the overall complexity of generating $MATCH^m(q_i^m)$ at Step 3^m requires comparisons of at most

$$\sum_{j=1}^{n} (\delta(i)^2 + \alpha(i)\lambda(i) + \beta(j)\eta(j) + P(i , j)) .$$

Range determination test is also needed when restriction is made at (3) of Step 4^m, i.e., ranges of the terms in the consequent of K_j^m and their corresponding variables in q^m need to be determined. Since the consequent of K_j^m and q^m have at most $\gamma(j)$ and ϵ atomic formulas with $*$, respectively, the test requires comparisons of no more than

$$\gamma(j) + \epsilon .$$

Thus the complexity of (3) of Step 4^m is at most

$$\sum_{m=1}^{\xi(j)} (\gamma(j) + \epsilon + C_m^j D_m^j).$$

Let N be $Max(N_1, N_2)$ where N_1 is the number of atomic formulas with * in q^m and N_2 is the largest possible number of atomic formulas with * in the antecedent ψ_j^m of any $K_j^m \in KB_{\Sigma H}^{mm}$. Then for any i, $1 \leq i \leq 2^r - 1$, $\alpha(i) \leq N$, for any j, $1 \leq j \leq n$, $\beta(j) \leq N$ and $\gamma(j) \leq N$, and $\epsilon \leq N$. The overall complexity of " $\longmapsto^m$ " is concluded as follows:

$$\sum_{i=1}^{2^r-1} [\sum_{j=1}^{n} (\delta(i)^2 + \alpha(i)\lambda(i) + \beta(j)\eta(j) + P(i, j)) + \sum_{j=1}^{n} \sum_{m=1}^{\xi(j)} (\gamma(j) + \epsilon + C_m^j D_m^j)]$$
$$\leq \sum_{i=1}^{2^r-1} \sum_{j=1}^{n} [L^2 + 2NK + M^K(2\log M^K + 1) + (2N + M^2)K].$$

It has been shown previously that the complexity of " $\longmapsto$" which corresponds to the preceding complexity of " $\longmapsto^m$ " is

$$\sum_{i=1}^{2^r-1} \sum_{j=1}^{n} [L^2 + M^K(2\log M^K + 1) + M^2 K].$$

Therefore, it is concluded that the complexity of " $\longmapsto^m$ " is augmented by

$$\sum_{i=1}^{2^r-1} \sum_{j=1}^{n} [4NK].$$

The preceding term signifies how much extra computation is entailed in " $\longmapsto^m$ " which is unnecessary in " $\longmapsto$." The amount of extra computation depends on the database, the queries and the knowledge about the data.

At the end of Section 6.2, it has been mentioned that $KB_{\Sigma H}$ is expanded by a class of equivalent axioms to the IA axioms in KB to enlarge the class of queries to be handled by the KBS. Finally, in the rest of the section it is illustrated how the

$KB_{\Sigma H}$ expanded by the equivalent axioms is applied to a query which otherwise may not be applied to. Suppose the user query q_1 in (5.2) had been equivalently given as q_2,

$$q_2 = \exists z\ \exists w\ \exists t^{\Sigma S}\ \exists x\ \exists y\ \exists v\ (\ It(z,w,t^{\Sigma S}) \cap Sa(x,y,z) \cap De(y,u,v)\ )$$

where $\forall x\ (S(x) \leftrightarrows (x = sedan \cup x = bus))$. Let q be the matrix of the existential closure of q_2,

$$q = It(z,w,t^{\Sigma S}) \cap Sa(x,y,z) \cap De(y,u,v)\ . \tag{7.4}$$

Then no subformula of q matches the IA axiom (7.2) although $It(z,w,t^{\Sigma S}) \cap Sa(x,y,z)$ of (7.4) "semantically" matches (7.2) in the sense that $DEF(DB(Auto), \exists w\ \exists t^{\Sigma S}\ It(z,w,t^{\Sigma S}) \cap Sa(x,y,z)) \subseteq DEF(DB(Auto), Sa(x,y,z^{\Sigma P}))$. However, the revised version, say q^*, of q in (7.4) can still be derived by using the IA axiom (6.11) which was previously shown equivalent to (7.2). First, (6.11) is Skolemized into

$$\begin{aligned}\forall y\ \forall x\ \forall z\ \forall w\ \forall t^{\Sigma R}\ (Sa(x,y,z) \cap It(z,w,t^{\Sigma R}) \rightarrow \\ De(y, g(y,x,z,w,t^{\Sigma R}), f^Q(y,x,z,w,t^{\Sigma R}))),\end{aligned} \tag{7.5}$$

where $g(y,x,z,w,t^{\Sigma R})$ and $f^Q(y,x,z,w,t^{\Sigma R})$ are the Skolem functions replaced for the variables u and $v^{\Sigma Q}$ of (6.11), respectively. Then similar procedure can be applied to (7.5) and (7.4), as had been done for (7.2) and the matrix of (7.1), to conclude q^*,

$$q^* = It(z,w,t^{\Sigma S}) \cap Sa(x,y,z) \cap De(y,u,w^{\Sigma Q})\ . \tag{7.6}$$

It is clear that (7.6) shows more precise URC's than (7.4).

7.2. Correctness of the Inference Procedure

In general, for any symbolic manipulation procedure designed to carry out inference, it must to be justified whether the result obtained syntactically is indeed valid semantically. For $q \in Q_c^m$, let $KB_{\Sigma H}^m ; q \longmapsto q^*$. What matters is whether the revised query q^* is equivalent to the original query q. In this section the issue of equivalence between the revised query and the original query is discussed.

As a preliminary step a lemma is first presented. The following notations are used in the lemma and elsewhere in this section: For a formula $\alpha(x_1, \cdots, x_n)$, let $<a_1, \cdots, a_n>$ stand for a variable assignment such that $\models_{DB} \alpha(x_1, \cdots, x_n)[a_1, \cdots, a_n]$. For such assignment $<a_1, \cdots, a_n>$, a_i is called an assignment element, or just an element, corresponding to x_i. Then by $<a_1, \cdots, a_n> | x_{i_j}$, $i_j \in \{i_1, \cdots, i_n\}$, it is meant the element a_{i_j} which corresponds to x_{i_j}. For a subformula $\beta(x_{i_1}, \cdots, x_{i_k})$ of $\alpha(x_1, \cdots, x_n)$, $\{i_1, \cdots, i_k\} \subseteq \{1, \cdots, n\}$, $<a_1, \cdots, a_n> | \beta$ stands for the subassignment $<a_{i_1}, \cdots, a_{i_k}>$ of $<a_1, \cdots, a_n>$ such that $\models_{DB} \beta(x_{i_1}, \cdots, x_{i_k})[a_{i_1}, \cdots, a_{i_k}]$.

<u>Lemma 7.2.1</u>

For $q \in Q_c^m$, let $KB_{\Sigma H}^m ; q \longmapsto q^*$. For some assignment $<a_1, \cdots, a_m>$, if $\models_{DB} q\,[a_1, \cdots, a_m]$ then there is an assignment $<a_1', \cdots, a_m'>$ satisfying $\models_{DB} q^*\,[a_1', \cdots, a_m']$.

Proof. The inference procedure "$\longmapsto$" is a process of revising the input query q to another form by applying each element in $KB_{\Sigma H}^m$ until q can no longer be revised. Therefore it suffices to show that for each $K_j \in KB^m_{\Sigma H}$ if $K_j \; ; q \;\; \longmapsto q^*$ then the lemma holds.

Let $K_j \; ; q \;\; \longmapsto q^*$. Without loss of generality, let $K_j{}'$ be of the following form which is the original form of K_j before it is Skolemized and its quantifiers are dropped:

$$K_j{}' = \forall x_1 \cdots \forall x_n \; (\psi_j(x_1, \cdots, x_n) \rightarrow \exists z_1 \cdots \exists z_k \; R(x_{a_1}, \cdots, x_{a_k}, z_1, \cdots, z_k)) ,$$

where $\{x_{a_1}, \cdots, x_{a_k}\} \subseteq \{x_1, \cdots, x_n\}$. Let q and q^* which both have m , $n + k \le m$, free variables be expressed as $q(y_1, \cdots, y_m)$ and $q^*(y_1^*, \cdots, y_m^*)$, respectively. Let some $q_i(y_{i_1}, \cdots, y_{i_n}) \in SUB(q)$, $\{y_{i_1}, \cdots, y_{i_n}\} \subseteq \{y_1, \cdots, y_m\}$, match K_j , i.e., q_i and ψ_j include an identical set of predicate symbols and $DEF(DB \; , q_i) \subseteq DEF(DB \; , \psi_j)$.

The proof is shown in a constructive way. For an assignment $<a_1, \cdots, a_m>$, let it hold that $\models_{DB} q \; [a_1, \cdots, a_m]$. Since q_i matches K_j , it follows that

$$\models_{DB} \; \psi_j \; [<a_1, \cdots, a_m> | \; q_i] \; .$$

Since K_j is true in DB , it follows that there is an assignment, say, $<d_1, \cdots, d_n, e_1, \cdots, e_k>$, satisfying

$$\models_{DB} \psi_j \cap R \; [d_1, \cdots, d_n, e_1, \cdots, e_k] , \text{ and}$$

$$<d_1, \cdots, d_n, e_1, \cdots, e_k> | \; \psi_j = <d_1, \cdots, d_n> = <a_1, \cdots, a_m> | \; q_i \; .$$

Let an atomic formula, say $R_q(y_{u_1}, \cdots, y_{u_k}, y_{v_1}, \cdots, y_{v_k})$, $\{y_{u_1}, \cdots, y_{u_k}, y_{v_1}, \cdots, y_{v_k}\} \subseteq \{y_1, \cdots, y_m\}$, in q and the consequent

$R(x_1, \cdots x_h, z_1, \cdots, z_k)$ in $K_j{}'$ constitute the restriction pair (R, R_q) associated with q_i and K_j. For some $\{y_{w_1}, \cdots, y_{w_l}\} \subseteq \{y_{v_1}, \cdots, y_{v_k}\}$ in R_q and some $\{z_{w_1}, \cdots, z_{w_l}\} \subseteq \{z_1, \cdots, z_k\}$ in R, let $Ran(y_{w_r}) \subset Ran(z_{w_r})$, $1 \leq r \leq l$, which therefore means q^* is obtained by restricting q by K_j in the following way: For each r, $1 \leq r \leq l$, substitute y_{w_r} by a variable v_r whose range $Ran(v_r) = Ran(z_{w_r})$.

Accordingly, a new assignment $<a_1', \cdots, a_m'>$ can be constructed from $<a_1, \cdots, a_m>$ and $<d_1, \cdots, d_n, e_1, \cdots, e_k>$ in the following way: For each r, $1 \leq r \leq l$, the element $<a_1, \cdots, a_m> | y_{w_r}$ in $<a_1, \cdots, a_m>$ is replaced by the element $<d_1, \cdots, d_n, e_1, \cdots, e_k> | z_{w_r}$. Let the resulting $<a_1, \cdots, a_m>$ be $<a_1', \cdots, a_m'>$. From the way q^* is obtained by restricting q by K_j and from the way $<a_1', \cdots, a_m'>$ is constructed from $<a_1, \cdots, a_m>$ and $<d_1, \cdots, d_n, e_1, \cdots, e_k>$, it follows that

$$\models_{DB} q^* [a_1', \cdots, a_m'] .$$

Q.E.D.

The preceding lemma can be said to signify the soundness of "$\longmapsto$" in the following sense: if q_c and q_c^* are the existential closures of q and q^*, respectively, then $KB_{\Sigma H}^m ; q \longmapsto q^*$ implies $KB_{\Sigma H}^m \cup \{q_c\} \models q_c^*$. The soundness of "$\longmapsto$" can be easily understood by the following argument: as long as it is known that all the dealers who are supplied cars are car dealers and that there are some dealers who have been supplied items $B47$, $V01$, or $V03$ which are cars, then it is valid to conclude that there are some dealers who are car dealers.

Lemma 7.2.1 is used in showing the equivalence of q' and q. The issue of the equivalence of the revised query q' and the original query q is directly related to why only the IA axioms having its corresponding FDA axioms in KB is included in $KB_{\Sigma H}$ when constructing $KB_{\Sigma H}$ from KB. Before showing their equivalence in a formal way, the role of FDA axioms in their equivalence is first illustrated in the following. Consider the functional dependency axiom in *DEALERS* from *d#* to *d_type* ,

$$\forall x \ \forall y \ \forall z \ \forall y' \ \forall z' (De(x,y,z) \cap De(x,y',z') \rightarrow z = z'), \tag{7.7}$$

and the IA axiom depicting a relationship between fractions of *SALES* and *DEALERS* ,

$$\forall y \ (\exists x \ \exists z^{\Sigma P} \ Sa(x,y,z^{\Sigma P}) \rightarrow \exists u \ \exists v^{\Sigma Q} \ De(y,u,v^{\Sigma Q})) \tag{7.8}$$

where P is meant by $\forall x \ (P(x) \leftrightarrows (x = B47 \cup x = V01 \cup x = V03 \cup x = W09))$ and Q , $\forall x \ (Q(x) \leftrightarrows (x = 50 \cup x = 51))$. It is not difficult to see that only when (7.7) is combined with (7.8), (7.8) is interpreted as "All the dealers who are supplied items $B47$, $V01$, $V03$ or $W09$ are exclusively car dealers". (7.8) alone only asserts "Any dealer who is supplied items $B47$, $V01$, $V03$ or $W09$ is a car dealer although that dealer may deal in other items". This implies that the IA axiom (7.8) requires the existence of (7.7) in the knowledge base to guarantee that the two fragments associated with the consequent of (7.8) are disjoint, i.e.,

$$DEF(DB(Auto), De(y,u,v^{\Sigma Q})) \cap DEF(DB(Auto), De(y,u,v^{\Sigma Q'})) = \phi$$

where $\forall x (Q'(x) \leftrightarrows \neg Q(x))$.

The above argument can be more realistically illustrated by entering the tuple $<$ 01A, Lansing, 80 $>$ in *DEALERS* in which case the functional dependency in

DEALERS from *d#* to *d_type* no longer exists. Let the database with the new tuple be $DB(Auto)'$. It is clear that (7.8) is still valid both in $DB(Auto)$ and in $DB(Auto)'$, but (7.7) is valid only in $DB(Auto)$. In this case, the following is clear:

$$DEF(DB(Auto)', De(y,u,v^{\Sigma Q})) \cap DEF(DB(Auto)', De(y,u,v^{\Sigma Q'}))$$
$$= \{< \text{01A} , \text{Lansing} , 80 >\} .$$

The preceding argument is formalized by the following theorem.

<u>Theorem 7.2.2</u> Equivalence of q and q^* .

For $q \in Q_c^m$, let $KB_{\Sigma H}^m ; q \longmapsto q^*$. Then $DEF(DB,q) = DEF(DB,q^*)$.

Proof. For the same reason stated at the beginning of the proof of Lemma 7.2.1, it suffices to show that for each $K_j \in KB_{\Sigma H}^m$ if $K_j ; q \longmapsto q^*$ then $DEF(DB,q) = DEF(DB,q^*)$.

Let $K_j ; q \longmapsto q^*$. Showing that $DEF(DB,q^*) \subseteq DEF(DB,q)$ is trivial, because q^* is a restricted version of q. Here it is only shown that $DEF(DB,q) \subseteq DEF(DB,q^*)$ holds.

Without loss of generality, let K_j' be of the following form which is the original form of K_j before it is Skolemized and its quantifiers are dropped:

$$K_j' = \forall x_1 \cdots \forall x_n \; (\psi_j(x_1, \cdots, x_n) \rightarrow \exists z_1 \cdots \exists z_k \; R(x_{a_1}, \cdots, x_{a_k}, z_1, \cdots, z_k)) ,$$

where $\{x_{a_1}, \cdots, x_{a_k}\} \subseteq \{x_1, \cdots, x_n\}$. Let q and q^* which both have m , $n + k \leq m$, free variables be expressed as $q(y_1, \cdots, y_m)$ and $q^*(y_1^*, \cdots, y_m^*)$, respectively. Let some $q_i(y_{i_1}, \cdots, y_{i_n}) \in SUB(q)$, $\{y_{i_1}, \cdots, y_{i_n}\} \subseteq \{y_1, \cdots, y_m\}$, matches K_j , i.e., q_i and ψ_j include an identical set of predicate symbols and

$DEF(DB, q_i) \subseteq DEF(DB, \psi_j)$. Let an atomic formula, say $R_q(y_{u_1}, \cdots, y_{u_k}, y_{v_1}, \cdots, y_{v_k})$, $\{y_{u_1}, \cdots, y_{u_k}, y_{v_1}, \cdots, y_{v_k}\} \subseteq \{y_1, \cdots, y_m\}$, in q and the consequent $R(x_1, \cdots x_h, z_1, \cdots, z_k)$ in K_j' constitute the restriction pair (R, R_q) associated with q_i and K_j. For some $\{y_{w_1}, \cdots, y_{w_l}\} \subseteq \{y_{v_1}, \cdots, y_{v_k}\}$ in R_q and some $\{z_{w_1}, \cdots, z_{w_l}\} \subseteq \{z_1, \cdots, z_k\}$ in R, let $Ran(y_{w_r}) \subset Ran(z_{w_r})$, $1 \leq r \leq l$, which therefore means q^* is obtained by restricting q by K_j in the following way: For each r, $1 \leq r \leq l$, substitute y_{w_r} by the variable v_r whose range $Ran(v_r) = Ran(z_{w_r})$.

Let the FDA axiom corresponding to K_j be of the form

$$\forall s_1 \cdots \forall s_h \forall t_1 \cdots \forall t_k \forall t_1' \cdots \forall t_k' \; (R(s_1, \cdots, s_h, t_1, \cdots, t_k) \cap R(s_1, \cdots, s_h, t_1', \cdots, t_k') \rightarrow t_{w_1} = t_{w_1}' \cap \cdots \cap t_{w_l} = t_{w_l}') \cdots (1)$$

where $\{w_1, \cdots, w_l\} \subseteq \{1, \cdots, k\}$. (1) states that there is a functional dependency in R from $s_1, \cdots, s_h$ to $t_{w_1}, \cdots, t_{w_l}$.

Let $<a_1, \cdots, a_m> \in DEF(DB, q)$. In order to prove that $DEF(DB, q) \subseteq DEF(DB, q^*)$ holds, it must be shown that $<a_1, \cdots, a_m> \in DEF(DB, q^*)$ holds. This is shown by contradiction. First, from the hypothesis $<a_1, \cdots, a_m> \in DEF(DB, q)$ it is implied that

$$\models_{DB} q\,[a_1, \cdots, a_m] \cdots (2).$$

By Lemma 7.2.1, (2) implies that there is an assignment $<a_1', \cdots, a_m'>$ satisfying

$$\models_{DB} q^*\,[a_1', \cdots, a_m'] \cdots (3).$$

Since q^* is a restricted version of q, (3) implies that

$$\models_{DB} q\,[a_1', \cdots, a_m'] \cdots (4).$$

To prove by contradiction, let $<a_1, \cdots, a_m> \notin DEF(DB, q^*)$ additionally hold. This additional hypothesis implies that

$$\not\models_{DB} q^* [a_1, \cdots, a_m] \cdots (5).$$

Now (3) and (5) are considered. From (3), (5) and that the way $<a_1', \cdots, a_m'>$ of (3) is constructed [see the proof of Lemma 7.2.1], the following holds:

$<a_1, \cdots, a_m>$ and $<a_1', \cdots, a_m'>$ are identical except that for some w_r, $w_r \in \{w_1, \cdots, w_l\}$, $<a_1, \cdots, a_m> | y_{w_r} \neq <a_1', \cdots, a_m'> | y_{w_r}$. $\cdots$ (6)

Now $<a_1', \cdots, a_m'>$ in (4) is considered. Let $<b_1', \cdots, b_h', c_1', \cdots, c_k'>$ be $<a_1', \cdots, a_m'> | R_q(y_{u_1}, \cdots, y_{u_h}, y_{v_1}, \cdots, y_{v_k})$. Then from (4) it follows that

$$\models_{DB} R_q [b_1', \cdots, b_h', c_1', \cdots, c_k'] \cdots (7).$$

Now $<a_1, \cdots, a_m>$ in (2) is considered. Let $<b_1, \cdots, b_h, c_1, \cdots, c_k>$ be $<a_1, \cdots, a_m> | R_q(y_{u_1}, \cdots, y_{u_1}, y_{v_1}, \cdots, y_{v_k})$. Then from (2) it follows that

$$\models_{DB} R_q [b_1, \cdots, b_h, c_1, \cdots, c_k] \cdots (8).$$

From (6), (7) and (8), the followings are concluded:

(i) $<b_1, \cdots, b_h> = <b_1', \cdots, b_h'>$, and

(ii) for some w_r, $w_r \in \{w_1, \cdots, w_l\}$, $<b_1, \cdots, b_h, c_1, \cdots, c_k> | y_{w_r} \neq$ $<b_1', \cdots, b_h', c_1', \cdots, c_k'> | y_{w_r}$.

(i) and (ii) implies that there is no functional dependency in R_q from $y_{u_1}, \cdots, y_{u_k}$ to $y_{w_1}, \cdots, y_{w_l}$. This further implies that there is no functional dependency in R from $s_1, \cdots, s_i$ to $t_{w_1}, \cdots, t_{w_l}$ since R_q and R are identical predicates. This fact contradicts (1). Thus it follows that $<a_1, \cdots, a_m> \in DEF(DB, q^*)$ that means $DEF(DB, q) \subseteq DEF(DB, q^*)$ holds. *Q.E.D.*

7.3. Horizontal Partitioning

In this section two issues are discussed: how the estimated *URC*'s are used for partitioning the relations; and how the partitions of the relations obtained by this approach should be interpreted.

The former issue is straightforward. The notion of a bipartition is first introduced: Given a revised query expression, say, q^*, let $R(q^*)$ be an atomic formula shown in q^*. Let *the bipartition of the relation* $R^{DB}(q^*)$ *obtained by* $R(q^*)$, denoted by $\Pi_b(R(q^*))$, be defined

$$\Pi_b(R(q^*)) = \{DEF(DB, R(q^*)), \overline{DEF}(DB, R(q^*))\}$$

where $\overline{DEF}(DB, R(q^*)) = R^{DB}(q^*) - DEF(DB, R(q^*))$.

When $KB^{\pi}_{\Sigma H} \cup q \longmapsto q^*$, the revised query expression q^* can then be viewed as a way of obtaining a bipartition of each relation referred to by q^*. That is, a relation being referred to by q^* is divided into two fragments, one part $DEF(DB, R(q^*))$ that is needed to answer q^* and the other $\overline{DEF}(DB, R(q^*))$ that is not needed. The set of these two fragments is conceived as a bipartition of the relation $R^{DB}(q^*)$. For instance, from the revised query expression q^* in (7.3)

$$q^* = Sa(x, y, z^{\Sigma I}) \cap De(y, u, w^{\Sigma Q}),$$

a bipartition of *DEALERS*, namely, *CAR_DEALERS* and *NON_CAR_DEALERS* and a bipartition of *SALES*, namely, *SALES_I* and *SALES_II* can be obtained. These two bipartitions are illustrated in Figure 7.2 of the following:

CAR_DEALERS

d#	address	d_type
01A	Ann Arbor	51
07A	Flint	50
55L	Flint	51

SALES_I

div#	d#	item#
01AP	01A	V01
02AP	01A	B47
04AP	01A	V03
05AP	55L	V03

↗ *DEALERS* ↘

↗ *SALES* ↘

NON_CAR_DEALERS

d#	address	d_type
03A	Dearborn	30
26M	Cleveland	20
33B	Cleveland	30
48B	Rockford	31
65B	Detroit	20
66L	Nile	23
70A	Lansing	70

SALES_II

div#	d#	item#
01AP	07A	W09
01PP	55L	S01
01PP	07A	P02
02PP	03A	P02
03PP	01A	P03
03PP	03A	S02
05PP	55L	S02

Figure 7.2 Bipartitions of the Relations *DEALERS* and *SALES* .

It is noticed that the bipartition of *DEALERS* is not derivable from the original query expression q_1 of (5.2),

$$q_1 = \exists x \ \exists y \ \exists z^{\Sigma I} \ \exists v \ (Sa(x,y,z^{\Sigma I}) \cap De(y,u,v)) .$$

In fact, more than one query can conceivably refer to each specific relation in the database. Let $Q^*(R)$ be the collection of the restricted versions of queries referring to the relation R^{DB}. One way to obtain a partition of the relation R^{DB} is to intersect the blocks of all the possible bipartitions each of which is obtained from a restricted query expression in $Q^*(R)$. In order to be more specific, the following notion is introduced: For two partitions Π^1 and Π^2 of the relation R^{DB}, $\Pi^1 \sqcap \Pi^2 \overset{d}{=} \{ S : S = B_i \cap B_j$ where $B_i \in \Pi^1$ and $B_j \in \Pi^2$, and $S \neq \phi \}$. Since the commutativity and the associativity hold for $\sqcap$, let $\Pi^1 \sqcap \cdots \sqcap \Pi^n$ be written by $\underset{i \in \{1, \ldots, n\}}{\sqcap} \Pi^i$. Formally stated, *the partition, denoted by* $\Pi(R)$, *of the relation* R^{DB} *obtained from the given* $Q^*(R)$ is

$$\Pi(R) = \underset{q^* \in Q^*(R)}{\sqcap} \Pi_b(R(q^*)) .$$

At a glance, the approach of intersecting all the possible bipartitions looks like a crude way of partitioning each relation in the database. However, this approach is meaningful in the sense that the partitions obtained from the revised query expressions by this approach is more refined than those obtained directly from the user provided query expressions. This further implies that when the fragments of the partitions are dispersed over the sites of a network, the fragments of the former partitions can be more flexibly distributed than those of the latter partitions. Various data allocation algorithms such as [MoLe77, IrKh79, Aper81, CeNW83] can be used to determine an optimal or suboptimal dispersion of the data by treating the fragments as the unit objects of distribution.

7.4. Conclusions and Future Work

A knowledge-based approach has been described in which *URC*'s are derived from the user queries to the database and the knowledge about the data. In order to describe the user queries and the knowledge, ordinary many-sorted language is extended. In this extended language, the user queries are expressed in a specific form, called Σ-normal form, and the knowledge useful for this purpose is identified by five types of axiom schemas. The knowledge is applied to each query expression via an inference mechanism to derive a revised query expression. From the revised query expressions, *URC*'s are estimated. Horizontal partitioning can be based on the estimated *URC*'s.

The work which has been shown so far can be further extended into three directions. One direction is to expand the knowledge base of the KBS by accommodating a larger class of knowledge. Possibly more knowledge is useful for the intended purpose. It can be represented in terms of different types of axiom schemas in L_Σ and, in such case, it is expected that the inference procedure ($\mid\!\longrightarrow$) may have to be modified. A more sophisticated inference procedure may be required.

The second direction is to study the problem of allocating the fragmented relations. Although the fragmented relations can be distributed over a network by adopting some of the currently known data allocation models, the allocation obtained from this approach may not reflect the logical intricacy among the fragments. That means, during the process of answering queries, the relationships among the fragments may not be used fully to reduce the unnecessary preselection or join operations that are the major benefits sought by distributing the partitioned fragments. This problem results because the currently known data allocation models do not take into account the logical relationships among the fragments as a design resource. A new data allocation model is needed that combines the relationships among the fragments with a quantitative optimization model.

The third direction is to investigate a distributed query optimization based on the knowledge base. When the fragments are dispersed over the network, the logical relationship among the fragments can guide various query processing strategies including how preselection operations can be dispensed with, how useless join operation can be eliminated, and how parallel distributed query processing can be scheduled over a network. When their relationships are complex enough, their role in guiding the process of answering queries can be more than what the conventional data directories usually do. The logical relationships can constitute a meta-knowledge base and it can be used in conjunction with the conventional data directory in an intelligent way to optimize processing the queries.

PART II

In this part, a type of problem is first identified which may occur when a resolution scheme is applied to many-sorted theory. In order to avoid such a problem, an extension of the first-order language called *one-sorted language with aggregate variables* is introduced. It is shown that any many-sorted theory can be converted into an equivalent theory in a one-sorted language with aggregate variables. Aggregate variables allow the introduction of range-restricted variables dynamically in the structure which is expanded by definitions. This allows the introduction of a new resolution scheme named *Unification over the Weakest Range* (or UWR-resolution). The completeness of UWR-resolution is shown and the efficiency of UWR-resolution is discussed.

CHAPTER VIII

A MANY-SORTED RESOLUTION BASED ON AN EXTENSION OF A ONE-SORTED LANGUAGE

8.1. Introduction

Within the field of automatic theorem proving, the advantages of many-sorted logic are well known [Haye71, Hens72, Cohn83]. A language of many-sorted logic offers more compact expressive power than the corresponding language of one-sorted logic, and so a theory is expressed with a much smaller number of shorter clauses in the former than in the latter. When a resolution scheme is used, the smaller number of shorter clauses means a shorter refutation. Furthermore, the refutation sequence is further shortened when the sortal information is used as a metaknowledge preventing irrelevant resolvents from being generated.

It was only recently that a theoretical foundation for many-sorted resolution was established by Walther [Walt83, Walt84a]. Walther presented a many-sorted calculus, called ΣRP-calculus, in which a resolution and the so-called weakening rule are employed as the inference rules of the system. He showed the completeness of ΣRP-calculus and also showed how the ΣRP-calculus is related to its corresponding one-sorted calculus. In his sequel paper, Walther also demonstrated the power of a many-sorted resolution by an example called "Schubert's streamroller" [Walt84b].

However, when Walther's approach is applied to a certain class of many-sorted theories, his approach still generates irrelevant resolvents which degrade the overall deductive efficiency. The many-sorted theories falling in this class are those satisfying a certain relationship among the sorts. By an example, an illustration is given of what this relationship is and what irrelevant resolvents are generated.

Example 8.1.1

Let x_b , x_c , x_d , and x_e be the variables ranging over the sorts B , C , D , and E , respectively, where $D \subset B$, $D \subset C$, $E \subset B$, and $E \subset C$. The theory to be refuted is given by:

(1) $\forall x_b \; (P(x_b) \cup \exists x_d \; Q(x_b , x_d))$,

(2) $\forall x_c \; \neg P(x_c)$,

(3) $\forall x_e \; \forall x_c \; \neg Q(x_e , x_c)$.

If (1) and (2) are chosen as parent clauses to be resolved, because x_b of $P(x_b)$ in (1) and x_c of $\neg P(x_c)$ in (2) are unifiable† over the sort D , and if (1) is expressed as $P(x_b) \cup Q(x_b , f^d(x_b))$ using a Skolem function $f^d(x_b)$ whose range is restricted to D , the two clauses can be resolved using a most general unifier (mgu) $\theta = \{ y_d/x_b , y_d/x_c \}$ where y_d ranges over the sort D . The resolvent then is

(4) $Q(y_d , f^d(y_d))$ (1)+(2) .

It is now seen that (4) cannot be resolved with any other clauses, not even with (3) because there is no sort known as a subsort of $D \cap E$. A dead end has been

† A variable v is *unifiable* with a term t *over* the sort S if there is a substitution θ that unifies $\{ v , t \}$, i.e., $v\theta = t\theta$, and the results of the instantiations $v\theta$ and $t\theta$ are both terms of sort S .

reached. The unsatisfiability can be shown either by resolving $P(x_b)$ in (1) and $\neg P(x_c)$ in (2) with a variable of sort E or by resolving $Q(x_b, f^d(x_b))$ in (1) and $\neg Q(x_e, x_c)$ in (3) with a variable of sort E ; (4) is a useless resolvent.

Generating the types of useless resolvents illustrated in the preceding example can be avoided. Had there been another sort $G = B \cap C$, (1) and (2) could have been unified over the sort G giving the resolvent

$$(4') \quad Q(x_g, f^d(x_g)) \qquad\qquad (1)+(2),$$

where the variable x_g ranges over the sort G . The clause $(4')$ can be resolved further with (3) over the sort E resulting in the empty clause $\Box$. There is no dead end here. There is a problem, however, that if the sort G is unavailable, the variable x_g cannot be introduced in the middle of the deduction. In an ordinary many-sorted language, a variable cannot be introduced unless the range of the variable agrees with any of the a priori fixed sorts, which is a common problem often caused by the inflexible usage of an ordinary many-sorted language†.

To alleviate such a situation of the preceding example, an extension is proposed of the one-sorted language called ***one-sorted language with aggregate variables*** (L_{Σ}^{1}) into which a many-sorted theory can be translated and which may be dynamically extended to bypass the problem illustrated previously. In Part I, aggregate variables were embedded into a many-sorted language resulting in the language called ***many-sorted language with aggregate variable*** . Here aggregate variables are embedded in a one-sorted language.

---

† Some discussion about the inflexible usage of many-sorted language is found in [Cohn83] in which Cohn suggested a way to improve the expressiveness of many-sorted logic.

Aggregate variables allow us the dynamic introduction of range-restricted variables without revising the a priori fixed structure. Using the dynamic range-restricting nature of the aggregate variables, an efficient many-sorted resolution scheme named *unification over the weakest range* (UWR-resolution) is presented, which is designed to avoid generating useless resolvents as illustrated in the preceding example. There are two issues to be discussed, the completeness of UWR-resolution and the efficiency of UWR-resolution.

8.2. Related Literature

In general, automatic theorem proof systems are divided into two classes: the systems belonging to the first class start with a given set of logical formulas and create new formulas by using certain inference rules until a refutation is concluded. The systems belonging to the second class do not create any new formulas but test certain conditions ensuring unsatisfiability of the given set of formulas. The former includes resolution-based proof system, and the latter includes mating-based proof systems such as Andrew's mating calculus [Andr81] or Bibel's matrix calculus [Bibe81]. Here the only concern is with the resolution-based proof systems.

In order to introduce some background for the resolution-based proof systems, the introductory statement by Davis and Putnam in [DaPu60] is quoted:

> "The hope that mathematical methods employed in the investigation of formal logic would lead to purely computational methods for obtaining mathematical theorems goes back to Leibniz and has been revived by Peano around the turn of the century and by Hilbert's school in the 1920's. Hilbert, noting that all of classical mathematics could be formalized within quantification, declared that the problem of finding an algorithm for determining whether or not a given formula of quantification theory is valid was the central problem of mathematical logic. And indeed, at one time it seemed as if investigations of this 'decision' problem were on the verge of success. However, it was shown by Church and by Turing that such an algorithm cannot exist. This result led to considerable pessimism regarding the possibility of using modern digital computers in deciding significant mathematical questions.

However, recently there has been a revival of interest in the whole question. Specifically, it has been realized that while no 'decision procedure' exists for quantification theory there are many proof procedures available · · · ."

An important contribution to the area of automatic theorem proving was made by Herbrand. Herbrand proposed in his thesis [Herb30] a deductive system that later turned out to be complete and far more efficient than other previously known deductive systems. The result, known as the Herbrand theorem, was later adopted further by many researchers and led to the invention of various proof procedures. Quine presented a proof procedure for quantification theory [Quin55], and Wang and Gilmore have each produced working programs that employ proof procedures in quantification theory. Although Quine's work was restricted to theoretic aspects of the proof procedure, Wang's and Gilmore's programs were actual working programs run on computing machines, which account for important initial contributions. Both Wang's and Gilmore's programs, however, were very inefficient due to the combinatorial explosion in determining the inconsistency of the given formula, although these methods are superior in many cases to truth table methods which are the crudest way of determining the inconsistency of the given formula. Both Wang's and Gilmore's programs run into difficulty with some fairly simple examples.

Wang's and Gilmore's methods were improved a few months after their results were published by Davis and Putnam [DaPu60]. Davis and Putnam proposed a new way of determining the inconsistency of the given formula while avoiding the problem of the type that occurred in Gilmore's program. However, their improvement was still not enough.

A major breakthrough was made by Robinson [Robi65a] who introduced the so-called "resolution principle." His resolution-based proof system was much more efficient than any earlier proof procedure. However, this system was still inefficient due to the many irrelevant and redundant formulas which were generated during the derivation of a refutation. Such pitfalls in Robinson's resolution triggered the creation of various refined forms of the resolution principle in the attempt to increase further its efficiency. Some of these refinements include hyper-resolution by Robinson [Robi65b], renameable resolution by Meltzer [Melt66], the set-of-support strategy by Wos, Robinson and Carson [WoRC65], all of which were later unified into semantic resolution by Slagle [Slag67]; lock resolution by Boyer [Boye71]; linear resolution, which was independently proposed by Loveland [Love70] and by Luckham [Luck70] and which was later strengthened by Anderson and Bledsoe [AnBl70], Reiter [Reit71], Loveland [Love72], and Kowalski and Kuehner [KoKu70]; and unit resolution by Wos, Carson, and Robinson [WoCR64] and Chang [Chan70].

Recently, researchers realized that the deductive efficiency of using the resolution can be improved significantly if the deduction is based on a many-sorted calculus along with incorporating the preceding types of refinements. Deduction based on a many-sorted calculus goes back to Herbrand [Herb30]. In his thesis Herbrand established the fact that the deduction based on a many-sorted logic is equivalent to the deduction based on its corresponding one-sorted logic. Since then, various forms of many-sorted calculus have been proposed and investigated by Schmidt [Schm38, Schm51], Wang [Wang52], Hailperin [Hail57], and Idelson [Idels64].

Several researchers suggested some practical theorem proving programs based on a many-sorted calculus without sound theoretical foundation [Weyh77, BoMo79]. It was only recently that a theoretical foundation for many-sorted resolution was reported by Walther and Cohn [Walt83, Cohn83, Walt84a]. Walther presented a many-sorted calculus based on resolution and paramodulation that is called ΣRP-calculus, and Cohn suggested a way to improve the expressiveness of a many-sorted logic in which a many-sorted resolution corresponding to Robinson's resolution is used as an inference rule. In his sequel paper, Walther demonstrated by an example the power of a many-sorted resolution [Walt84b].

8.3. Organization

The rest of Part II is organized in a way similar to Part I. In Chapter IX, L_{Σ}^{1} is introduced: syntax of L_{Σ}^{1}, interpretation of L_{Σ}^{1}, and the Σ-extensibility of L_{Σ}^{1}. L_{Σ}^{1} then is used as the language for describing a many-sorted theory.

In Chapter IX, it is first shown how a many-sorted theory is formalized in L_{Σ}^{1}. It is clarified that the only concern is with a certain class of many-sorted theories. The problem that was illustrated by an example in Section 8.1 is then formally described.

In Chapter IX, the UWR-resolution is introduced and the completeness of the UWR-resolution is shown. In order to prove the completeness of UWR-resolution, in Section 11.2, the L_{Σ}^{1}-version of the Herbrand theorem is introduced.

Finally in Chapter IX, the issues about the efficiency of the UWR-resolution are discussed. To discuss the efficiency, a hypothetic many-sorted resolution is introduced, namely Σ-resolution, that does not employ the technique of introducing a new sort dynamically as the resolution is being carried out. The efficiency of the UWR-resolution is then measured by comparing the refutation of a given many-sorted theory generated by the UWR-resolution with that generated by the Σ-resolution.

In Appendix B, some intermediate steps needed to introduce the L_{Σ}^{1}-version of the Herbrand theorem are shown. In Appendix C, two complete refutations are shown that show the inconsistency of an example many-sorted theory. One is generated by the UWR-resolution and the other, by the Σ-resolution. In Appendix D, two alternative approaches are given which embody the idea of unifying a pair of variables satisfying a certain condition over the weakest possible range: (i) an approach in which the theory in a many-sorted language L_m is repeatedly translated into *a revised language of* L_m along the way the refutation of the theory is carried out, and (ii) an approach in which the theory to be refuted is expressed in a generalized version of an ordinary many-sorted language whose variable sets and constant sets are not necessarily disjoint. In Appendix E, it is shown that the generalized version of an ordinary many-sorted language which was introduced in Appendix D is as legitimate as the ordinary many-sorted language.

CHAPTER IX

ONE-SORTED LANGUAGE WITH AGGREGATE VARIABLES L^1_Σ

9.1. Syntax of L^1_Σ

Aggregate variables can be embedded in a one-sorted language as well as in a many-sorted language. When the aggregate variables are embedded in the former, it is called *a one-sorted language with aggregate variables* (L^1_Σ). L^1_Σ is the special case of L_Σ where there is only one sort. In this sense formal introduction of L^1_Σ is unnecessary. Nevertheless, for the sake of clarification and for the purpose of letting Part II stand alone, L^1_Σ is fully introduced in this chapter. Syntax of L^1_Σ is first given in this section.

Two types of variables are available in a one-sorted language with aggregate variables L^1_Σ: simple variables and *aggregate variables*. A simple variable of L^1_Σ is the same as the ordinary variable of a one-sorted language. An aggregate variable is syntactically an ordinary sort variable, but semantically a variable whose range of interpretation is restricted by a unary relation rather than to a sort domain. Formally stated, an *aggregate variable* is of the form $x^{\Sigma P_i}$ in which $x^{\Sigma P_i}$ ranges over the unary relation indicated by the unary predicate symbol P_i .

Let J , L , and K be, respectively, a relation index set, a function index set and a constant index set. In addition, let I be an index set for some unary relations. Let λ and ξ be functions such that $\lambda : J \rightarrow N^+$ and $\xi : L \rightarrow N^+$ where N^+ is the set of positive integers.

Definition 9.1.1

A *one-sorted language with aggregate variables* L_Σ^1 then consists of the following: (1) parentheses (,) ; (2) constant symbol C_k for each $k \in K$; (3) simple variables x_1 , $\cdots$, x_m , $\cdots$, and aggregate variables $x_1^{\Sigma P_i}$, $\cdots$, $x_n^{\Sigma P_i}$, $\cdots$, for each $i \in I$, where P_i is a unary predicate symbol; (4) a $\lambda(j)$-ary predicate symbol R_j for each $j \in J$; (5) a $\xi(l)$-ary function symbol F_l for each $l \in L$; (6) logical connectives $\neg$ and $\rightarrow$; and (7) a universal quantifier $\forall$. •

When it is convenient, L_Σ^1 is represented as a quintuple, $L_\Sigma^1 = < P , R , F , C , \rho >$ where P is a unary predicate set whose members are exclusively used in the superscripts of aggregate variables, R is a predicate symbol set, F is a function symbol set, C is a constant symbol set and ρ is the arity function such that $\rho : R \cup F \rightarrow N^+$, where N^+ is the set of positive integers.

Based on this language, the *terms* of L_Σ^1 are defined as usual except that each variable is now either a simple variable or an aggregate variable. The set of *atomic formulas* of L_Σ^1, $Atom(L_\Sigma^1)$, is also defined as usual. The set of *well-formed formulas* of L_Σ^1, $Form(L_\Sigma^1)$, is then defined recursively as: (i) if $\alpha \in Atom(L_\Sigma^1)$, then $\alpha \in Form(L_\Sigma^1)$; (ii) if $\alpha, \beta \in Form(L_\Sigma^1)$, then so are $\neg \alpha$, $(\alpha \rightarrow \beta)$, and $\forall v\ \alpha$ where v is either a simple variable or an aggregate variable ; (iii) nothing else, except the expressions obtained by finite applications of (i) and (ii), is in $Form(L_\Sigma^1)$. The definable syntactic objects $\cup$, $\cap$, $\leftrightarrows$ and $\exists$, and the standard notions such as *sentences* are also introduced in the usual way.

9.2. Interpretation of L_Σ^1

A structure is needed to interpret each formula in L_Σ^1 . Let OS_a be a structure for L_Σ^1 . Then $OS_a = <\Omega, \{\dot{P}_i\}_{i\in I}, \{\dot{R}_j\}_{j\in J}, \{\dot{F}_l\}_{l\in L}, \{\dot{C}_k\}_{k\in K}>$† where Ω is the universe of OS_a ; $\dot{P}_i$ is a unary relation $\dot{P}_i \subseteq \Omega$; $\dot{R}_j$ is a $\lambda(j)$-ary relation $\dot{R}_j \subseteq \Omega^{\lambda(j)}$; $\dot{F}_l$ is a $\xi(l)$-ary function $\dot{F}_l : \Omega^{\xi(l)} \to \Omega$; and a distinguished element $\dot{C}_k$ is an element of Ω .

The interpretation of a formula in the structure OS_a then requires a variable assignment function s as follows:

Definition 9.2.1

For set V of variables of L_Σ^1 and the universe Ω of structure OS_a , s is an assignment function $s : V \to \Omega$ such that for a simple variable x , $s(x) = a$, where $a \in \Omega$; for an aggregate variable $x^{\Sigma P_i}$, $s(x^{\Sigma P_i}) = a$, where $a \in \dot{P}_i$. ●

Assignment function for the terms of L_Σ^1 is defined as usual. For notational convenience symbol s is also used for the assignment for the terms. The validity of each formula is determined by the following interpretation rules.

† From the next section on " · " is omitted on a symbol as long as the meaning of the symbol is unambiguous.

Definition 9.2.2

For $R_j(t_0, \cdots, t_{\lambda(j)})$, ψ_1, $\psi_2 \in Form(L_\Sigma^1)$, where t_i's are terms, the satisfaction of the formulas with respect to s in OS_a is defined by,

(1) $\models_{OS_a} R_j(t_0, \cdots, t_{\lambda(j)})[s]$ iff $<s(t_0), \cdots, s(t_{\lambda(j)})> \in R_j$,

(2) $\models_{OS_a} \neg \psi [s]$ iff $\not\models_{OS_a} \psi [s]$,

(3) $\models_{OS_a} \psi_1 \rightarrow \psi_2 [s]$ iff if $\models_{OS_a} \psi_1 [s]$ then $\models_{OS_a} \psi_2 [s]$,

(4) For a simple variable x , $\models_{OS_a} \forall x \ \psi [s]$ iff for any $a \in \Omega$, $\models_{OS_a} \psi [s(x \mid a)]$, and

(5) For an aggregate variable $x^{\Sigma P_i}$, $\models_{OS_a} \forall x^{\Sigma P_i} \psi [s]$ iff for any $a \in \dot{P}_i$, $\models_{OS_a} \psi [s(x^{\Sigma P_i} \mid a)]$,

where for variables v_m and v_k , $s(v_m \mid a)(v_k) = \begin{cases} s(v_k) & \text{if } v_m \neq v_k \\ a & \text{if } v_m = v_k \end{cases}$. ●

As a corollary to the definition, the interpretations of $\cup$, $\cap$, $\leftrightarrows$ and $\exists$ can also be easily defined.

9.3. Σ-Extensibility of L_Σ^1

Two results are shown in this section: (i) how a formula in a many-sorted language is translated into L_Σ^1, and (ii) how in L_Σ^1 variables ranging over a priori undefined sorts can be introduced while the structure associated with L_Σ^1 is expanded by definitions.

As a preliminary step to showing the former result, a many-sorted language (L_m) is formally defined first. A *many-sorted language* L_m *with sort index set* I consists of the followings: (1) $|I|$ infinite disjoint sets $V^1, \cdots, V^{|I|}$ where the elements of V^i, $1 \leq i \leq |I|$, are called variables of sort i; (2) $|I|$ disjoint sets $C^1, \cdots, C^{|I|}$ where the elements of C^i, $1 \leq i \leq |I|$, are called constant symbols of sort i; (3) for each n-tuple $<i_1, \cdots, i_n>$, $\{i_1, \cdots, i_n\} \subseteq I$, a set $R^{<i_1, \cdots, i_n>}$ whose elements are called predicate symbols of sort $<i_1, \cdots, i_n>$; (4) for each $n+1$-tuple $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, a set $F^{<i_1, \cdots, i_n, i_{n+1}>}$ whose elements are called function symbols of sort $<i_1, \cdots, i_n, i_{n+1}>$; (5) logical connectives $\neg$ and $\rightarrow$; and (6) a universal quantifier $\forall$.

For the sort index set I, let there be a partial order relation $S \subseteq I \times I$, called *sort ordering,* such that $< i_j, i_k > \in S$ if and only if sort i_j *is a subsort of* sort i_k. For each $i \in I$, let $SUB(i) \stackrel{d}{=} \{ i_p : < i_p, i > \in S \}$. The syntax rule of L_m with respect to the sort ordering S is given in the following†. First, the set of *terms of sort* i is inductively defined as follows: (i) any variable of sort i or constant symbol of sort i is a term of sort i, and (ii) if f is a function symbol of sort $<i_1, \cdots, i_n, i_{n+1}>$ and $t_1, \cdots, t_n$ are terms of sort $i_1^p, \cdots, i_n^p$, respectively, where $i_j^p \in SUB(i_j)$, $1 \leq j \leq n$, then $f(t_1, \cdots, t_n)$ is a term of sort i_{n+1}. The set of *atomic formulas* of L_m is defined to be of the form $A(t_1^p, \cdots, t_n^p)$ where A is an n-place predicate symbol of sort $<i_1, \cdots, i_n>$ and t_j^p, $1 \leq j \leq n$, is a term of sort $i_j^p \in SUB(i_j)$. The set of *well-formed formulas* of L_m is then defined as usual.

† The syntax rule of L_m given here is similar to that of a many-sorted language given by [Walt83]. Similar syntax rule is also mentioned in [Wang52] as a more general form than the syntax rule of a many-sorted language given in [Ende72, KrKr67].

Definable symbols $\cup$, $\cap$, $\leftrightarrows$ and $\exists$ are introduced in L_m as usual and the interpretation of the formulas of L_m is also given as usual.

Now it is shown how a formula in L_m is translated into L_Σ^1 . Let σ_m be a formula in L_m . When σ_m is translated into L_Σ^1, let the translated formula be denoted by σ_Σ^*. If a sort variable, say x_i of sort $i \in I$, occurs in σ_m such as

$$\sigma_m = \forall x_i \text{ \_\_\_\_\_\_ } x_i \text{ \_\_\_\_\_\_ },$$

then in σ_Σ^* x_i is replaced by an aggregate variable, say $x^{\Sigma P_i}$, i.e.,

$$\sigma_\Sigma^* = \forall x^{\Sigma P_i} \text{ \_\_\_\_\_\_ } x^{\Sigma P_i} \text{ \_\_\_\_\_\_ },$$

where P_i is introduced as the corresponding unary predicate symbol to sort i . If a function symbol, say f of sort $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, occurs in σ_m such as

$$\sigma_m = \text{ \_\_\_\_\_\_ } f \text{ \_\_\_\_\_\_ },$$

then in σ_Σ^* f is superscripted with $P_{i_{n+1}}$, i.e.,

$$\sigma_\Sigma^* = \text{ \_\_\_\_\_\_ } f^{P_{i_{n+1}}} \text{ \_\_\_\_\_\_ },$$

where $P_{i_{n+1}}$ is introduced as the corresponding unary predicate symbol to sort i_{n+1} . When $n = 0$, the preceding function symbol translation includes how a constant symbol in σ_m is translated into L_Σ^1 , i.e., if c is a constant symbol of sort i that occurred in σ_m , then c is superscripted with the unary predicate, say P_i , that corresponds to sort i , i.e., c^{P_i} . The preceding translation of σ_m into σ_Σ^* implies that L_Σ^1 is as convenient as L_m in abbreviating the relativized expressions in a one-sorted language into more compact forms.

For convenience, let L_m with sort index I be a quadratuple $L_m = < R , F , C , \rho >$ where R is a predicate symbol set, F is a function symbol set, C is a constant symbol set, and ρ is the arity function such that $\rho : R \cup F \rightarrow N^+$ where N^+ is the positive integer set. Then the language L_Σ^1 for σ_Σ^* is a quintuple $L_\Sigma^1 = < P , R , F' , C' , \rho >$. P in L_Σ^1 is a unary predicate symbol set whose elements are the unary predicate symbols that are introduced during the translation of σ_m into σ_Σ^*, for instance, such as P_i in $x^{\Sigma P_i}$ and P_{i_n+1} in $f^{P_{i_n+1}}$. F' and C' in L_Σ^1 are the function symbol set and the constant symbol set, respectively, whose members are obtained by superscripting appropriately their respective function symbols and constant symbols in F and C .

As far as semantics for the formula σ_Σ^* is concerned, the structure for L_Σ^1, say $OS_a(L_\Sigma^1)^*$, can be constructed from the many-sorted structure for L_m , say $MS(L_m)$. Let $MS(L_m)$ be a quadratuple $MS(L_m) = < \{S_i\}_{i \in I} , \dot{R} , \dot{F} , \dot{C} >$ where I is the sort index set. Then $OS_a(L_\Sigma^1)^*$ is a quintuple $OS_a(L_\Sigma^1)^* = < \Omega , \dot{P} , \dot{R} , \dot{F}' , \dot{C} >$ where $\Omega = \bigcup_{i \in I} S_i$, $\dot{P} = \{ P_i$: for each $i \in I$, S_i is assigned to $P_i \}$ and $\dot{F}' = \{ f'$: for each function $f \in \dot{F}$, $f : S_{i_1} \times \cdots \times S_{i_n} \rightarrow S_{i_n+1}$, f' is an arbitrary extension of f, $f' : \Omega^n \rightarrow \Omega \}$. The following theorem is shown for the translation of σ_m into σ_Σ^*:

<u>Theorem 9.3.1</u>

A sentence σ_m in L_m is true in $MS(L_m)$ iff σ_Σ^* in L_Σ^1 is true in $OS_a(L_\Sigma^1)^*$.

Proof. Let the sets of terms of L_m and L_Σ^1 be denoted by $Term(L_m)$ and $Term(L_\Sigma^1)$, respectively. Let s be an assignment function $s : Term(L_m) \rightarrow \bigcup_{i \in I} S_i$. Then along with the translation of σ_m into σ_Σ^*, there can be defined an assignment function s^*, $s^* : Term(L_\Sigma^1) \rightarrow \Omega$, such that $s^*(t^*) = s(t)$ where t^* stands for the translation of $t \in Term(L_m)$ into L_Σ^1.

Proof is shown by induction on the length of σ_m. First, let σ_m be an atomic formula of the form $R(t_1, \cdots, t_n)$ where $t_1, \cdots, t_n \in Term(L_m)$. Let the relations designated by R in $MS(L_m)$ and in $OS_a(L_\Sigma^1)^*$ be $R^{MS(L_m)}$ and $R^{OS_a(L_\Sigma^1)^*}$, respectively. Then $R^{MS(L_m)} = R^{OS_a(L_\Sigma^1)^*}$. From the way that s^* is defined, it follows that

$$\models_{MS(L_m)} R(t_1, \cdots, t_n)[s] \iff < s(t_1), \cdots, s(t_n) > \in R^{MS_a(L_m)}$$
$$\iff < s^*(t_1^*), \cdots, s^*(t_n^*) > \in R^{OS_a(L_\Sigma^1)^*}$$
$$\iff \models_{OS_a(L_\Sigma^1)^*} R(t_1^*, \cdots, t_n^*)[s^*].$$

Since $\sigma_\Sigma^* = R(t_1^*, \cdots, t_n^*)$, the theorem holds when σ_m is atomic.

Suppose the theorem holds for all formulas of length less than or equal to h. Inductive step must be shown for the formulas of length $h+1$. When the formulas of length $h+1$ is obtained from the formulas of length less than or equal to h by using $\neg$ or $\rightarrow$, the proof is trivial. Only the following inductive step is shown. Let $\sigma_{m,h}$ be a formula in L_m of length h and let $\sigma_{\Sigma,h}^*$ be the translation of $\sigma_{m,h}$ into L_Σ^1. Induction hypothesis implies that for any assignment function s and its corresponding assignment function s^*, $\models_{MS(L_m)} \sigma_{m,h}[s]$ iff $\models_{OS_a(L_\Sigma^1)^*} \sigma_{\Sigma,h}^*[s^*]$. Let σ_m be $\forall x_i\ \sigma_{m,h}$ where x_i is a variable of sort i. It follows that

$$\models_{\overline{MS}(L_m)} \sigma_m\ [s] \iff \models_{\overline{MS}(L_m)} \forall x_i\ \sigma_{m,h}\ [s]$$

$$\iff \text{for any } a \in S_i\ ,\ \models_{\overline{MS}(L_m)} \sigma_{m,h}\ [s(x_i \mid a)]$$

by the induction hypothesis,

by the way the translation is made, and

by the way $OS_a(L_\Sigma^1)^*$ is constructed from $MS(L_m)$

$$\iff \text{for any } a \in P_i\ ,\ \models_{OS_a(L_\Sigma^1)^*} \sigma_{\Sigma,h}^*\ [s^*(x^{\Sigma P_i} \mid a)]$$

by Definition 9.2.2 (5)

$$\iff \models_{OS_a(L_\Sigma^1)^*} \forall x^{\Sigma P_i}\ \sigma_{\Sigma,h}^*\ [s^*]$$

Since $\sigma_\Sigma^* = \forall x^{\Sigma P_i}\ \sigma_{\Sigma,h}^*$, the theorem holds for the formulas of length $h+1$. *Q.E.D.*

The preceding theorem assures that any formula in L_m can be translated into L_Σ^1 only by using aggregate variables. It may well be assumed from here on that a many-sorted theory can be expressed in L_Σ^1 only by using aggregate variables. In addition to showing the expressive power of L_Σ^1 , the preceding theorem suffices to justify the validity of embedding ***aggregate variables*** in a one-sorted language.

The power of L_Σ^1 over L_m lies in the fact that in the former ***a variable whose range is restricted to any subset of the universe*** Ω ***can be introduced as needed in its extension,*** whereas in the latter a sort variable ranging over an a priori undefined sort may not be introduced. This means that one of the problems of L_m , namely, the inflexible usage of sort variables (e.g., [Cohn83]), can now be overcome. How the inflexible usage is overcome is explained in detail in the rest of this section.

Now it is shown how in L_Σ^1 variables ranging over new sorts that have not been defined a priori can be introduced while the structure associated with L_Σ^1 is expanded by definitions. Let a theory T_o in a one-sorted language be equivalently expressed as a many-sorted theory, say T_m, in L_m. Let the language for T_m be $L_m(T_m)$ †. Let x_1 and x_2 be the sort variables of $L_m(T_m)$ which range over the sorts S_1 and S_2, respectively. An inflexible usage of sort variables is displayed when another formula in a one-sorted language, say a logical consequence ϕ_o of T_o,

$$\phi_o = \forall x \ (S_1(x) \cap S_2(x) \ \rightarrow \ \psi(x)) \tag{9.1}$$

needs to be further abbreviated in $L_m(T_m)$. If a new variable ranging over $S_1 \cap S_2$, say x_k, can be introduced, ϕ_o of (9.1) can be abbreviated to $\forall x_k \ \psi(x_k)$ in $L_m(T_m)$. Unless the sort equal to $S_1 \cap S_2$ has been defined in the sort structure for $L_m(T_m)$, however, doing so requires the revision of the sort structure for $L_m(T_m)$ to accommodate the sort equal to $S_1 \cap S_2$. Compared with this, in L^1_Σ in order to introduce a variable ranging over a previously undefined set, the structure for L^1_Σ only needs to be expanded by Σ-definition.

Let T_Σ be the translation of T_m into L^1_Σ. Let $L^1_\Sigma(T_\Sigma)$ be the language for T_Σ. Let $x^{\Sigma S_1}$ and $x^{\Sigma S_2}$ be two aggregate variables of L^1_Σ that range over the relations S_1 and S_2, respectively. In order to introduce a variable ranging over $S_1 \cap S_2$, all that must be done is to add a new unary predicate symbol S_k to $L^1_\Sigma(T_\Sigma)$, abbreviate ϕ_o by

$$\forall x^{\Sigma S_k} \ \psi(x^{\Sigma S_k}), \tag{9.2}$$

and augment T_Σ by the defining axiom

$$\forall x \ (S_k(x) \leftrightarrows S_1(x) \cap S_2(x)).$$

† By the language of T_m, it is meant the language whose variables are those of L_m and whose relations and function symbols are those which occur in T_m.

The extended language, say $L_{\Sigma}^{1}{}'$, is formally called a Σ-*extension* of L_{Σ}^{1} and the augmented theory, say T_{Σ}' , a Σ-*extension* of T_{Σ}.

As far as the semantics of the new predicate symbols in $L_{\Sigma}^{1}{}'$ are concerned, such as S_k of (9.2), their corresponding unary relations must be introduced in the structure for L_{Σ}^{1}. Suppose OS_a is a model of T_{Σ}. In a way similar to the one shown in Lemma 3.3.2 of Part I, it can be shown that there is a unique expansion by definition of OS_a , say OS_a' , which is a model of T_{Σ}' . More specifically, OS_a' is called ***an expansion by Σ-definition*** of OS_a .

Let the characteristic of L_{Σ}^{1} that allows a more compact expressive power in its extended language be called Σ-*extensibility* of L_{Σ}^{1}. The validity of Σ-extensibility of L_{Σ}^{1} can be shown in a way similar to the one that shows the validity of Σ-extensibility of L_{Σ} in Theorem 3.3.2 of Part I.

CHAPTER X

PROBLEM FORMULATION

10.1. Representation of a Many-Sorted Theory in L_Σ^1

The many-sorted theories of concern here are those which fall in a certain class. In this section, it is shown how these many-sorted theories can be described in L_Σ^1. Let T_m be a many-sorted theory expressed in a L_m with sort index set I and its associated sort ordering S which is a partial order relation in I. Let $L_m(T_m)$ be the language for T_m. Let $F^{<i_1, \cdots, i_n, i_{n+1}>}$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, stand for the function symbol set of sort $<i_1, \cdots, i_n, i_{n+1}>$ in $L_m(T_m)$. Corresponding to $L_m(T_m)$, let a L_Σ^1 be defined as shown in Section 9.3.3. For each $i \in I$, the L_Σ^1 has a unary predicate Q_i.

Let T_m be translated into L_Σ^1 and let the translated theory be denoted by T_Σ. Two facts must be included in T_Σ: each sort indicated by $i \in I$ is not empty and each function indicated by $f \in F^{<i_1, \cdots, i_n, i_{n+1}>}$ is well regulated over the corresponding sorts. These two facts can be described in L_Σ^1 in the following forms of axiom schemas: Let x, $x_1, \cdots, x_n$ be simple variables of L_Σ^1. Then

(i) for each $i \in I$, $\exists x \; Q_i(x)$, and

(ii) for each function symbol f in $L_m(T_m)$ of sort $<i_1, \cdots, i_n, i_{n+1}>$

$\forall x_1 \cdots \forall x_n \; (Q_{i_1}(x_1) \rightarrow \cdots \rightarrow Q_{i_n}(x_n) \rightarrow Q_{i_{n+1}}(f(x_1, \cdots, x_n)))$. When $n = 0$, the preceding formula becomes the sentence $Q_i(c)$ which indicates c is a constant symbol of sort i .

For the preceding types of axioms, however, their presence in T_Σ does not need to be stated explicitly. Since the facts described by the preceding two types of axioms hold for every many-sorted theory described in L^1_Σ, their presence can be simply assumed without their explicit inclusion. For example, when a resolution principle is applied to T_Σ, its refutation can be preceded under the assumption that the preceding types of axioms are implicit in T_Σ.

It can be said that many-sorted theories in general contain two types of nonlogical axioms, namely, type I nonlogical axioms and type II nonlogical axioms, that characterize each specific many-sorted theory. Type I nonlogical axioms are those that describe the relationships among the sorts. The type I nonlogical axioms can be expressed in the following form of schema in L^1_Σ:

$$\forall x\ (\ Q_{i_j}(x) \rightarrow Q_{i_k}(x)\ ) \tag{10.1}$$

where x is a simple variable and $< i_j\ , i_k > \in S$. The type II nonlogical axioms are any formulas of L^1_Σ.

The goal of this section is to show how the many-sorted theories concerned in this work are formalized by using the language L^1_Σ and an additional symbol " $\sqsubseteq$ " which is introduced shortly. Although the many-sorted theories concerned here can be expressed solely in L^1_Σ, for convenience the symbol " $\sqsubseteq$ " is additionally used. First, it is discussed that when applying a resolution principle to a many-sorted theory T_Σ, the two types I and II of nonlogical axioms of T_Σ can be expressed independently by using two different representation schemes. When a resolution principle is applied to T_Σ, deductions made using the type I nonlogical axioms of T_Σ are distinguished from deductions made using the type II nonlogical axioms of T_Σ. The deductions made from the former are relationships among the predicate symbols $\{Q_\iota\}_{\iota \in I}$ in L_Σ^1 and the deductions made from the latter are the resolvents of a set of clauses in L_Σ^1 which are generated by using the deductions made from the former exclusively as a metaknowledge (how this is done will be clear in Section 11.3 where the WR-unification algorithm is introduced). Such distinction between the two types of deductions implies that the two types of nonlogical axioms of T_Σ can be expressed independently by using two different representation schemes, one for the type I nonlogical axioms and the other for the type II nonlogical axioms.

It is discussed how the many-sorted theories concerned here are formalized by using the symbol " $\subseteq$ " and L_Σ^1. It is first shown how the symbol " $\subseteq$ " is used to express type I nonlogical axioms of the many-sorted theories. It was shown that the type I nonlogical axioms of a many-sorted theory include the instances of the schema (10.1). Let the symbol " $\subseteq$ " be used to indicate that

$$Q_{\iota_j} \subseteq Q_{\iota_k} \tag{10.2}$$

if and only if $\forall x \, (\, Q_{\iota_j}(x) \rightarrow Q_{\iota_k}(x) \,)$. An expression of the form (10.2) is called an *ordering axiom.* Let OA (acronym of *ordering axioms*) be a set of expressions of the form (10.2). It is clear that the type I nonlogical axioms of T_Σ can be expressed in terms of OA.

Showing how the type II nonlogical axioms of a many-sorted theory are expressed in L_Σ^1 is straightforward. Previously by Theorem 9.3.1 it has been shown that any formula in L_m can be expressed in L_Σ^1. Let T_Σ be a set of formulas in

L^1_Σ . Then from Theorem 9.3.1 it is clear that all the type II nonlogical axioms of a many-sorted theory can be expressed in terms of T_Σ . In conclusion, it is said that a many-sorted theory concerned in this work is formalized by an ordered pair $< OA , T_\Sigma >$.

The following is an example of the formalization of a many-sorted theory which falls in the class of many-sorted theories concerned here:

Example 10.1.1

The many-sorted theory in Example 8.1.1 can be expressed by an ordered pair $< OA , T_\Sigma >$ as follow:

OA †: (1) $D \subseteq B$, $D \subseteq C$,

(2) $E \subseteq B$, $E \subseteq C$,

T_Σ : (3) $\forall x^{\Sigma B} (P(x^{\Sigma B}) \cup \exists x^{\Sigma D} \; Q(x^{\Sigma B} , x^{\Sigma D}))$,

(4) $\forall x^{\Sigma C} \; \neg P(x^{\Sigma C})$,

(5) $\forall x^{\Sigma E} \; \forall x^{\Sigma C} \; \neg Q(x^{\Sigma E} , x^{\Sigma C})$.

T_Σ in the previously formalized $< OA , T_\Sigma >$ is a collection of formulas of L^1_Σ . In the rest of this section it is shown how the T_Σ is equivalently expressed as a collection of "clauses". First a few notions are introduced that is used throughout the rest of Part II.

Literals: For any $\alpha \in Atom(L^1_\Sigma)$, α is a literal and $\neg \alpha$ is also a literal.

Complements: For any $\alpha \in Atom(L^1_\Sigma)$, α is a complement of $\neg \alpha$ and also

† For simplicity, any ordering axiom of the form $Q_{i_j} \subseteq Q_{i_k}$ is omitted in OA . This convention is used throughout the Part II.

$\neg\, \alpha$ is a complement of α. The two literals α and $\neg\, \alpha$ are, in either order, a complementary pair.

Clauses: A finite set (possibly empty) of literals is a clause. A disjunction of literals is used as synonymous with a set of literals. The empty clause is denoted by $\square$.

Ground literals: A literal that contains no variables is a ground literal.

Ground Clauses: A clause whose each member is a ground literal is a ground clause. In particular, $\square$ is a ground clause.

Expressions: Terms and literals are the only expressions.

Now the Skolemization of the formulas in T_Σ is considered. For each $\psi \in T_\Sigma$ of a many-sorted theory $< OA \ , T_\Sigma >$, ψ can be transformed into a prenex normal form where the matrix contains no quantifiers and the prefix is a sequence of quantifiers. The matrix, since it does not contain quantifiers, can be transformed in a conjunctive normal form. Let the formula ψ be transformed into

$$Q_1 x_1 \cdots Q_n x_n \ M \tag{10.2}$$

where M is in a conjunctive normal form and Q_i, $1 \leq i \leq n$, is either $\forall$ or $\exists$. If (10.2) were a one-sorted formula, what is known as a Skolem normal form of (10.2) is obtained by the following: beginning with x_1, replace each existentially quantified variable in M, say x_r, $1 \leq r \leq n$, by a function $f(x_{s_1}, \cdots, x_{s_m})$, $1 \leq s_1 < \cdots < s_m < r$, and delete $Q_r x_r$ from the prefix.

When (10.2) is a formula in L^1_Σ, a modification is made to this Skolemization process. That is, each Skolemized function that is introduced in place of an existentially closed variable is superscripted with a unary predicate symbol that is accompanied with the variable. For instance, if x_r is replaced by a function $f(x_{s_1}, \cdots, x_{s_m})$ and x_r is an aggregate variable accompanied with a unary predicate symbol, say R, then x_r is replaced by the function $f^R(x_{s_1}, \cdots, x_{s_m})$.

Once each $\psi \in T_\Sigma$ is transformed into a Skolem normal form, the prefix of ψ is made implicit since it consists of only the universal quantifiers. After the prefix is dropped from ψ, ψ is a conjunction of clauses. Let a conjunction of clauses be used as synonymous with a set of clauses. In the rest of Part II, by a many-sorted theory it is meant an ordered pair $< OA , T_\Sigma >$ in which T_Σ is a set of clauses. An example follows:

Example 10.1.2

Consider Example 10.1.1. T_Σ in the $< OA , T_\Sigma >$ below is now a set of clauses. In clause (3), $f^D(x^{\Sigma B})$ is a Skolem function that is replaced for $x^{\Sigma D}$:

$$
\begin{array}{lll}
OA : & (1) & D \Subset B , \ D \Subset C , \\
 & (2) & E \Subset B , \ E \Subset C , \\
T_\Sigma : & (3) & P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, f^D(x^{\Sigma B})) , \\
 & (4) & \neg P(x^{\Sigma C}) , \\
 & (5) & \neg Q(x^{\Sigma E}, x^{\Sigma C}) .
\end{array}
$$

Finally, a few notations are introduced that are used in the rest of the Part II. First two defined symbols that are denoted by $\not\Subset$ and $\sqsubset$, respectively, are introduced. Associated with the symbol $\Subset$, the two symbols $\not\Subset$ and $\sqsubset$ are defined, respectively, as follows: $Q_{i_j} \not\Subset Q_{i_k}$ if and only if it is not the case that $Q_{i_j} \Subset Q_{i_k}$, and $Q_{i_j} \sqsubset Q_{i_k}$ if and only if $Q_{i_j} \Subset Q_{i_k}$ and $Q_{i_j} \neq Q_{i_k}$.

Now a unary predicate that corresponds to each term is introduced. The set of clauses T_Σ in a $< OA\ , T_\Sigma >$ is considered. Let $Term(T_\Sigma)$ be the set of all the terms which occur in T_Σ. For each term $t \in Term(T_\Sigma)$, a unary predicate symbol is determined by the outermost symbol of t. That is, if the outermost symbol of t is a function symbol of the form f^{P_i}, a variable of the form $x^{\Sigma P_i}$, or a constant of the form c^{P_i}, then P_i is the predicate symbol determined by the outermost symbol of t. Here P_i is called the unary predicate that corresponds to the term t. Such unary predicate symbol P_i of the preceding is denoted by $Ran(t)$ from here on.

Now the ordering axiom set OA of the $< OA\ , T_\Sigma >$ is considered. Let $t_i\ , t_j \in Term(T_\Sigma)$. In the rest of Part II, statements of the following form are often needed to be mentioned:

$$Ran(t_i) \sqsubseteq Ran(t_j) \in OA\ . \tag{10.3}$$

When OA is fixed, the statements of the form (10.3) can be made without explicitly mentioning OA. For notational simplicity, in the rest of Part II, $Ran(t_i) \sqsubseteq Ran(t_j)$ is used to mean that $Ran(t_i) \sqsubseteq Ran(t_j) \in OA$. Accordingly, by $Ran(t_i) \sqsubset Ran(t_j)$ it would often mean that $Ran(t_i) \sqsubseteq Ran(t_j) \in OA$ but $Ran(t_i) \neq Ran(t_j)$, and by $Ran(t_i) \not\sqsubseteq Ran(t_j)$, neither $Ran(t_i) \sqsubseteq Ran(t_j) \in OA$ nor $Ran(t_i) = Ran(t_j)$.

10.2. Finitely Many Most General Unifiers

The problem identified in Example 8.1.1, namely, the generation of useless resolvents that lead to dead ends, occurs only when a certain class of many-sorted theories is refuted by a resolution scheme. For instance, for the many-sorted theories with the tree structure stated in [Walt84a], this problem would never occur. When the tree constraint is lifted, however, this problem may appear. In this section, the conditions under which such problems may arise are formalized, this time in terms of L_Σ^1.

In general, when the resolution principle is applied to a many-sorted theory some restrictions are required in its unification procedure. In order to describe the restrictions more specifically, the following notion is introduced: A many-sorted theory $< OA , T_\Sigma >$ is considered. Let $L_\Sigma^1(T_\Sigma)$ mean the language of T_Σ †. Let P be the unary predicate symbol set of $L_\Sigma^1(T_\Sigma)$. Given the set P, let *a set of immediate predecessors* of a unary predicate symbol $P_i \in P$, denoted by $IM^P(P_i)$, be defined by $IM^P(P_i) \stackrel{d}{=} \{ P_j \mid P_j \in P , P_j \Subset P_i$ and if there is a $P_l \in P$ such that $P_j \subseteq P_l \subseteq P_i$, then $P_j = P_l$ or $P_l = P_i \}$. For simplicity, from here on the superscript P in the notation $IM^P(\cdot)$ is omitted. It can be done because if the theory $< OA , T_\Sigma >$ is given the unary set P is fixed. The restrictions are then: a variable v can be unified with a nonvariable term t iff $Ran(t) \subseteq Ran(v)$ and a variable v_i can be unified with a variable v_j iff $IM(Ran(v_i)) \cap IM(Ran(v_j)) \neq \phi$. The former restriction can be enforced easily in a many-sorted resolution by restricting that each substitution component t/v should satisfy the condition $Ran(t) \subseteq Ran(v)$. One way to incorporate the latter restriction in a many-sorted resolution would be the following‡: if there is a unary predicate symbol $S_k \in P$ such that $S_k \Subset Ran(v_i)$ and $S_k \Subset Ran(v_j)$, and, at the same time, there is no $S_l \in P$ satisfying $S_k \Subset S_l \Subset Ran(v_i)$ and $S_k \Subset S_l \Subset Ran(v_j)$, then $\{ v_i , v_j \}$ is unifiable

† By $L_\Sigma^1(T_\Sigma)$ it is meant the language whose variables are those of L_Σ^1 and whose relations and function symbols are those which occur in the set T_Σ of formulas. This notation is used in the rest of Part II.

‡ In [Walt83], a similar idea of incorporating the latter restriction was implemented by the inference rule called *"weakening rule"*.

with a substitution $\theta = \{ x_k / v_i , x_k / v_j \}$, where x_k is an aggregate variable accompanying with the predicate symbol S_k .

When the preceding method of incorporating the restrictions is directly implemented in a unification procedure, a certain situation arises that is called *generation of finitely many most general unifiers* (here only the case having a finite number of sorts is considered). The situation of generating finitely many mgus arises when two to-be-unified variables, say v_i and v_j , satisfy the following conditions:

(i) $Ran(v_i) \not\subseteq Ran(v_j)$ and $Ran(v_j) \not\subseteq Ran(v_i)$,

(ii) $| IM(Ran(v_i)) \cap IM(Ran(v_j)) | > 1$.

In fact, if x_k is a variable such that $Ran(x_k) = P_k$ and $P_k \in IM(Ran(v_i)) \cap IM(Ran(v_j))$, then any substitution $\theta = \{ x_k / v_i , x_k / v_j \}$ is a legitimate mgu of $\{ v_i , v_j \}$, since $v_i \theta = v_j \theta$. This implies that there are possibly as many mgus for $\{ v_i , v_j \}$ as $| IM(Ran(v_i)) \cap IM(Ran(v_j)) |$. For example, consider Example 10.1.2. when the ordering axioms are $D \subseteq B$, $D \subseteq C$, $E \subseteq B$ and $E \subseteq C$, $| IM(Ran(x^{\Sigma B})) \cap IM(Ran(x^{\Sigma C})) | = \{D , E\}$. Two different mgus are available for $\{x^{\Sigma B} , x^{\Sigma C}\}$, namely, $\{x^{\Sigma D} / x^{\Sigma B} , x^{\Sigma D} / x^{\Sigma C}\}$ and $\{x^{\Sigma E} / x^{\Sigma B} , x^{\Sigma E} / x^{\Sigma C}\}$ which both are legitimate mgus. As has been demonstrated in Example 8.1.1, the problem in this situation is that multiple resolvents can be derived from given two clauses and not all of them are indeed useful for the generation of the empty clause. In Section 11.1, a way to remedy this situation is formally proposed.

CHAPTER XI

UWR-RESOLUTION

11.1. Unification over the Weakest Range

First, a few basic notions are introduced that are needed for formal description of the resolution scheme called unification over the weakest range (UWR-resolution). These notions are concerned with the operation of instantiation, i.e., substitution of terms for variables in the clauses of L_Σ^1 .

A many-sorted theory $< OA \ , T_\Sigma >$ is considered. Given the ordering axiom set OA and the language $L_\Sigma^1(T_\Sigma)$, the following notions are introduced. Any expression of the form t/v where v is a variable and t is a term different from v satisfying $Ran(t) \subseteq Ran(v)$ is a ***wr-substitution component***. For two variables v_i and v_j of $L_\Sigma^1(T_\Sigma)$ that satisfy the conditions (i) $Ran(v_i) \not\subseteq Ran(v_j)$ and $Ran(v_j) \not\subseteq Ran(v_i)$ and (ii) $| IM(Ran(v_i)) \cap IM(Ran(v_j)) | > 1$, a pair denoted by $\{ t/v_i \ , t/v_j \}$ is a ***wr-subpair,*** if

(1) t/v_i and t/v_j are wr-substitution components where t is a new variable in $L_\Sigma^1(T_\Sigma)$,

(2) $L_\Sigma^1(T_\Sigma)$ is extended by including a new unary predicate symbol, say P_t , with $Ran(t) = P_t$, and

(3) OA is augmented so that for each unary predicate symbol $Q \in IM(Ran(v_i)) \cap IM(Ran(v_j))$, $(Q \sqsubseteq P_t) \in OA$, and $(P_t \sqsubseteq P_{v_i}) \in OA$ and $(P_t \sqsubseteq P_{v_j}) \in OA$ where P_{v_i} and P_{v_j} are the predicates indicated by $Ran(v_i)$ and $Ran(v_j)$, respectively.

A finite set (possibly empty) of wr-substitution components that possibly contain one or more wr-subpairs and none of the variables of which are same is a *wr-substitution*. In particular, ϵ denotes empty substitution. The notions such as *instantiation* and *composition of substitutions* are defined as usual. If E is an expression and θ is a wr-substitution, then the instantiation of E by θ is denoted by $E\theta$. If λ is also a wr-substitution, the composition of θ and λ is denoted by $\theta\lambda$.

A wr-substitution θ is called a *wr-unifier* for a set $\{ E_1, \cdots, E_n \}$ of expressions if and only if $E_1\theta = E_2\theta = \cdots = E_k\theta$. The set $\{ E_1, \cdots, E_k \}$ is said to be *unifiable* if there is a wr-unifier for it. A wr-unifier σ for a set $\{ E_1, \cdots, E_n \}$ of expressions is a *most general wr-unifier* (wr-mgu) if and only if for each wr-unifier θ for the set there is a wr-substitution λ such that $\theta = \sigma\lambda$. A *wr-resolvent* is a resolvent that is generated by using a wr-substitution as a unifier (this notion is defined in a more formal way in Section 11.3).

The Σ-extensibility of L^1_Σ plays the central role in introducing wr-subpairs. From the definition of a wr-subpair, it is clear that wr-resolvents are not expressible in the current vocabulary of L^1_Σ. They can only be expressed in an extended language of L^1_Σ. This idea is illustrated in the following example.

Example 11.1.1

The many-sorted theory $< OA \ , T_\Sigma >$ in Example 10.1.2 is considered.

$$
\begin{array}{lll}
OA : & (1) & D \sqsubseteq B \ , \ D \sqsubseteq C \ , \\
 & (2) & E \sqsubseteq B \ , \ E \sqsubseteq C \ , \\
T_\Sigma : & (3) & P(x^{\Sigma B}) \cup Q(x^{\Sigma B} \ , f^D(x^{\Sigma B})) \ , \\
 & (4) & \neg P(x^{\Sigma C}) \ , \\
 & (5) & \neg Q(x^{\Sigma E} \ , x^{\Sigma C}) \ .
\end{array}
$$

An example of a wr-resolvent is the following: For $x^{\Sigma B}$ of $P(x^{\Sigma B})$ in (3) and $x^{\Sigma C}$ of $\neg P(x^{\Sigma C})$ in (4), a wr-subpair $\{ x^{\Sigma K}/x^{\Sigma B} \ , x^{\Sigma K}/x^{\Sigma C} \}$ can be introduced if L^1_Σ is extended by a unary predicate symbol, say K, where $\forall x \ (K(x) \leftrightarrows B(x) \cap C(x))$. The extension of L^1_Σ requires $< OA \ , T_\Sigma >$ to be extended also. That is, upon introducing K, OA is augmented to $OA^{+'}$ by the ordering axioms as follows:

$$
(2^+) \quad K \sqsubseteq B \ , \ K \sqsubseteq C \ , \ D \sqsubseteq K \ , \ E \sqsubseteq K \ .
$$

Here the wr-subpair $\{ x^{\Sigma K}/x^{\Sigma B} \ , x^{\Sigma K}/x^{\Sigma C} \}$ itself is a wr-unifier of (3) and (4). Therefore the wr-resolvent of (3) and (4) that is generated by using $\{ x^{\Sigma K}/x^{\Sigma B} \ , x^{\Sigma K}/x^{\Sigma C} \}$ as a unifier is $Q(x^{\Sigma K} \ , f^D(x^{\Sigma K}))$. In the following, a refutation of the $< OA \ , T_\Sigma >$ is shown:

$$
\begin{array}{lll}
(6) & Q(x^{\Sigma K} \ , f^D(x^{\Sigma K})) \ , & (3)+(4) \\
(7) & \square \ . & (5)+(6)
\end{array}
$$

The above refutation shows that the wr-resolvent (6) of (3) and (4) is resolved with (5) resulting in $\square$.

So far the syntactic notion of the UWR-resolution has been introduced. Before ending this section, in the rest, the semantic notion of the UWR-resolution is discussed in terms of the structure associated with $L^1_\Sigma(T_\Sigma)$. When the outermost symbol of t is a function symbol, say f, (if the outermost symbol is a 0-place function symbol, then t is a constant) the unary relation indicated by $Ran(t)$ is *the codomain* of the function indicated by f. When the outermost symbol of t is a variable, t itself is a variable and the unary relation indicated by $Ran(t)$ is *the range* of the variable t. Let the unary relation indicated by $Ran(t)$ be denoted by $\dot{Ran}(t)$. It is not difficult to see that the range of the variable t in a wr-subpair $\{ t/v_i , t/v_j \}$ [i.e., $\dot{Ran}(t) = \dot{Ran}(v_i) \cap \dot{Ran}(v_j)$] is the weakest range over which $\{ v_i , v_j \}$ can be unified -- weakest in the sense that if $\dot{P}_v = \dot{Ran}(t)$, then there is no other unary relation $\dot{P}_l$ such that $\dot{P}_v \subset \dot{P}_l$ and $\{ v_i , v_j \}$ is still unifiable over $\dot{P}_l$. For this reason, the unification stated here is called **unification over the weakest range** and the resolution involving such unification is called **UWR-resolution.** The idea behind UWR-resolution is therefore to subsume all the possible unifications by one unification over the weakest possible range.

11.2. Herbrand Theorem for L^1_Σ Clauses

As a preliminary step to proving the completeness of UWR-resolution, in this section a modified version of the Herbrand theorem [Herb30] that is called the L^1_Σ-version Herbrand Theorem is presented. The L^1_Σ-version Herbrand theorem is used as the basis for proving the completeness of the UWR-resolution in the following section. This modified version of the Herbrand theorem is needed for two reasons: first, the Herbrand theorem is originally based on a one-sorted predicate calculus, but here a many-sorted predicate calculus is dealt with, and second, the original version of the Herbrand theorem cannot be used directly for proving the completeness of a resolution scheme, although it provides the theoretic basis for doing so.

In the modification, the original version of the Herbrand theorem is used as the starting point. First, based on that original version, a many-sorted version of the Herbrand theorem is established that is applicable to the clauses in an ordinary many-sorted language (L_m). Then, the many-sorted version applicable to the clauses in L_m is converted into another many-sorted version that is, this time, applicable to the clauses in L_Σ^1. Here the former step is of no concern as long as one such many-sorted version can be found in the literature. Such a version is given by Kreisel and Krivine [KrKr67] which they call "the uniformity theorem for predicate calculus with several types of variables." Thus the only concern here is to convert the many-sorted version of the Herbrand theorem by Kreisel and Krivine into another many-sorted version that suits our purpose.

Converting the many-sorted version by Kreisel and Krivine into the many-sorted version that suits our purpose consists of two steps: first, to convert the former into an intermediate version that is applicable to the L_Σ^1 clauses, and second, to convert the intermediate version into the form of the Herbrand theorem that can be directly used for proving the completeness of UWR-resolution. The first conversion step is straightforward and, therefore, is shown in Appendix B. In Appendix B, the following form of the L_Σ^1-version Herbrand theorem is derived as an intermediate result:

Theorem 11.2.1

Let $A(x_1, \cdots, x_n)$ be a quantifier free formula with free variables $x_1, \cdots, x_n$. Then $\forall x_1 \cdots \forall x_n\, A(x_1, \cdots, x_n)$ is unsatisfiable if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$, $1 \leq i \leq p$, of n-tuples of terms of $L_\Sigma^1(A)$† such that $A_1 \cap \cdots \cap A_p$ is unsatisfiable where A_i is obtained by replacing x_j, $1 \leq j \leq n$, in A by $t_j^{(i)}$.

This form of the Herbrand theorem further needs to be modified so that it can be directly used for proving the completeness of UWR-resolution. In the rest of this section it is shown how the further modifications are made.

First it is seen what is meant by Theorem 11.2.1. This theorem says that there is a procedure verifying the inconsistency of a prenex formula, say ψ. The formula ψ' is constructed from ψ which, being universal, can be written in the form $\forall x_1 \cdots \forall x_n\, A(x_1, \cdots, x_n)$ where A is quantifier free. Then formulas are generated of the form, for some $k > 0$,

$$A(t_1^{(1)}, \cdots, t_n^{(1)}) \cap \cdots \cap A(t_1^{(k)}, \cdots, t_n^{(k)}), \qquad (11.1)$$

where $t_j^{(i)}$'s are terms of $L_\Sigma^1(A)$. Each formula of this form is tested in a finite number of steps to determine whether or not it is inconsistent by using a truth table, i.e., by treating each atomic formula in (11.1) as a propositional variable. Then ψ is inconsistent if and only if an inconsistent formula of the form (11.1) is found.

† $L_\Sigma^1(A)$ stands for the language of A. By the language of a formula, it is meant the language whose variables are those of L and whose relations and function symbols are those which occur in formula A.

As the preceding procedure indicates, the Herbrand theorem provides a theoretical basis for the existence of a proof procedure for a quantification theory (strictly speaking, what is described is a refutation procedure rather than a proof procedure). The Herbrand theorem, however, does not address the details about how an actual proof procedure should look, for example, how terms are to be substituted for variables and how the inconsistency of the resulting formula of the form (11.1) can be checked. For developing an actual proof procedure, the most critical issue is how these two detailed processes can be made in a systematic way, since what matters in the actual proof procedure is the efficiency. Most of the proof procedures known today tackle this issue in one way or the other.

The preceding issue was first addressed by Quine [Quin55]. In his paper Quine presented two proof procedures called "method A" and "method B." In method A, he suggested a way to substitute terms for variables. Given a Skolemized normal form ψ, let a class of terms, say C, contain, to begin with, all those constants of ψ (or 'a', arbitrarily, if there is none). Further, if a non-zero-degree function symbol occurs in ψ, then the function, with members of C in place of the function's argument position(s), in turn belongs to C. This class C, usually infinite, which Quine called "the lexicon of ψ", is then the only set of terms that are substituted for the variables in ψ. This method of substituting terms for variables is restrictive in the sense that no such restriction is mentioned in the original version of the Herbrand theorem. The restrictive substitution, however, does not hamper the completeness of the proof procedure which is based on such restrictive substitution. Here the restriction is not necessary but only used as a technical aid for carrying out the substitutions. Quine's restrictive substitution strategy was later used in various machine based proof procedures [Gilm60, Robi65a].

In method B, Quine further suggested a methodology with which the inconsistency of a set of formulas can be proved without formulating a conjunction of all the formulas in the set. He also showed that doing this is more efficient than doing otherwise.

These two ideas, restrictive substitutions and proving the inconsistency of a set of formulas without formulating a conjunction of all the formulas in the set, later led to a specific form of the Herbrand theorem by Robinson [Robi65a]. Robinson used this version in proving the completeness of his resolution principle. The goal of this section is to derive a modification of the Robinson's version which is applicable to the clauses in L_Σ^1. The modified version is derived in the rest of this section.

First, two notions are introduced that are often called "Herbrand universe" and "saturation." The notion of Herbrand universe of a set of clauses in L_Σ^1 is given first: Let T_Σ be a set of clauses in L_Σ^1. For some index set I, let $\{P_i\}_{i \in I}$ be the unary predicate set† of $L_\Sigma^1(T_\Sigma)$. Let $MIN(\{P_i\}_{i \in I})$ be the subset of $\{P_i\}_{i \in I}$ such that if $P_j \in MIN(\{P_i\}_{i \in I})$ then for no $P_l \in MIN(\{P_i\}_{i \in I})$ is $P_l \Subset P_j$. Let F be the set of all function symbols that occur in T_Σ. For each $P_k \in MIN(\{P_i\}_{i \in I})$, if F contains a zero-degree function symbol, say c, such that $Ran(c) = P_k$, then the *functional vocabulary* of T_Σ is F; otherwise the functional vocabulary is the set $\{c\} \cup F$ where c is a constant symbol arbitrarily chosen to satisfy $Ran(c) = P_k$. Then *the Herbrand universe of* T_Σ is the set of all ground terms in which there occur only symbols in the functional vocabulary of T_Σ.

† The unary predicate set of L_Σ^1 is the set of unary predicates which accompany the aggregate variables of L_Σ^1.

Now the notion of saturation is the following: Let T_Σ be any set of clauses in L_Σ^1 and let P_Σ be any set of ground terms. Then *the saturation of* T_Σ *over* P_Σ, denoted by $P_\Sigma(T_\Sigma)$, is the set of all ground clauses obtainable from the clauses of T_Σ by replacing each variable, say v_i, in a clause of T_Σ with each member, say t_j, of P_Σ which satisfies the condition $Ran(t_j) \subseteq Ran(v_i)$ [occurrences of the same variable in any one clause is replaced by the same term].

The two preceding notions are illustrated by the following example:

Example 11.2.1

Consider the $< OA , T_\Sigma >$ of Example 10.1.2:

$$
\begin{array}{lll}
OA: & (1) & D \subseteq B, \; D \subseteq C, \\
 & (2) & E \subseteq B, \; E \subseteq C, \\
T_\Sigma: & (3) & P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, f^D(x^{\Sigma B})), \\
 & (4) & \neg P(x^{\Sigma C}), \\
 & (5) & \neg Q(x^{\Sigma E}, x^{\Sigma C}).
\end{array}
$$

The unary predicate set, say UP, of $L_\Sigma^1(T_\Sigma)$ is $\{B, C, D, E\}$. $MIN(UP)$ is then $\{D, E\}$. Let d^D and e^E be the constants such that $Ran(d^D) = D$ and $Ran(e^E) = E$. The functional vocabulary of T_Σ is then $\{d^D, e^E\} \cup \{f^D\}$. The Herbrand universe of T_Σ is the following:

$$\{d^D, e^E, f^D(d^D), f^D(e^E), f^D(f^D(d^D)), f^D(f^D(e^E)), \cdots ,\}.$$

Let P_Σ be a finite subset of the Herbrand universe of T_Σ, say $P_\Sigma = \{e^E, f^D(e^E)\}$. The saturation of T_Σ over P_Σ, $P_\Sigma(T_\Sigma)$, is the following:

$$
\begin{aligned}
P_\Sigma(T_\Sigma) = \{ & P(e^E) \cup Q(e^E, f^D(e^E)), \; P(f^D(e^E)) \cup Q(f^D(e^E), f^D(f^D(e^E))), \\
& \neg P(e^E), \; \neg P(f^D(e^E)), \; \neg Q(e^E, e^E), \; \neg Q(e^E, f^D(e^E)) \}.
\end{aligned}
$$

Notice that $\neg Q(f^D(e^E), e^E)$ and $\neg Q(f^D(e^E), f^D(e^E))$ are not included in $P_\Sigma(T_\Sigma)$, since $Ran(f^D(e^E)) \not\subseteq Ran(z^{\Sigma E})$ [notice that $Ran(f^D(e^E)) = D$].

When the two notions, Herbrand universe and saturation, are used, Theorem 11.2.1 can be rephrased in the form given below. It is assumed that for a many-sorted theory $< OA , T_\Sigma >$, OA is any *finite set* of ordering axioms and T_Σ is any *finite set* of clauses.

<u>Theorem 11.2.2</u>

A $< OA , T_\Sigma >$ is unsatisfiable if and only if *some finite subset* of $H(T_\Sigma)$ is unsatisfiable where H is the Herbrand universe of T_Σ.

Finally, with a little exercise of imagination, the preceding form of the Herbrand theorem can be further rephrased in the following form:

<u>Theorem 11.2.3</u> Herbrand Theorem for L^1_Σ Clauses.

A $< OA , T_\Sigma >$ is unsatisfiable if and only if for *some finite subset* P_Σ of the Herbrand universe of T_Σ, $P_\Sigma(T_\Sigma)$ is unsatisfiable.

The preceding theorem is the final L^1_Σ-version of the Herbrand theorem that was to be derived. In the following section this theorem is used for proving the completeness of UWR-resolution.

11.3. Completeness of UWR-Resolution

In this section the completeness of UWR-resolution is proved. This proof closely follows the proof of Robinson's resolution principle presented in [Robi65a]. First the completeness of UWR-resolution at the ground level must be established. At the ground level, however, there is no difference between the resolution of one-sorted ground clauses and the resolution of many-sorted ground clauses. For example, when two ground clauses are to be resolved, what must be determined is whether the two clauses contain a complementary pair of ground literals. In making such a decision, it is immaterial that each term in a ground clauses in L_Σ^1 is associated with a certain unary predicate symbol which is determined by the outermost symbol of the term. In the following, therefore, the ground resolution theorem for UWR-resolution is presented without its proof† as a modification of the ground resolution theorem for Robinson's resolution principle.

First, a few basic notions are introduced. Let C and D be two ground clauses or, as defined synonymously, two sets of ground literals. Let $L \subseteq C$ and $M \subseteq D$ be two singletons whose respective members form a complementary pair of ground literals. Then the ground clause $(C - L) \cup (D - M)$ is called *a ground resolvent of* C and D. Let T_Σ^g be any set of ground clauses. Then *the ground resolution of* T_Σ^g, denoted by $R(T_\Sigma^g)$, is the set of ground clauses consisting of the members of T_Σ^g and all ground resolvents of all pairs of members of T_Σ^g. *The* n^{th} *ground resolution of* T_Σ^g, denoted by $R^n(T_\Sigma^g)$, is defined for each $n \geq 0$ as follows: $R^0(T_\Sigma^g) = T_\Sigma^g$ and for $n \geq 0$, $R^{n+1}(T_\Sigma^g) = R(R^n(T_\Sigma^g))$.

---

† Formal proof can be found in [Robi65a].

Theorem 11.3.1 Ground Resolution Theorem for UWR-Resolution.

A $< OA , T_{\Sigma}^{g} >$ is unsatisfiable if only if $R^{n}(T_{\Sigma}^{g})$ contains $\Box$ for some $n \geq 0$.

By using this theorem, the Herbrand theorem for L_{Σ}^{1} clauses, Theorem 11.2.3, can be rephrased as follows:

Theorem 11.3.2

A $< OA , T_{\Sigma} >$ is unsatisfiable if only if for some finite subset P_{Σ} of the Herbrand universe of T_{Σ} and some $n \geq 0$, $R^{n}(P_{\Sigma}(T_{\Sigma}))$ contains $\Box$.

The rest of this section is devoted to showing how the preceding theorem leads to the theorem for the completeness of UWR-resolution. As the first step, the procedure known as unification algorithm must be given which shows how a mgu is derived for a set of clauses satisfying a certain condition. First, the notion of a disagreement set is introduced as usual. The disagreement set of a nonempty set W of expressions (excluding literals with negation symbol) is obtained by locating the first symbol (counting from the left) at which not all the expressions in W have exactly the same symbol and then extracting from each expression in W the subexpression that begins with the symbol occupying that position. The set of these respective subexpressions is the *disagreement set of W*.

In the following an algorithm is introduced which embodies the idea presented in Section 11.1, i.e., unifying two variables which satisfy a certain condition over the weakest range. This algorithm is called WR-unification algorithm and is applicable to any finite nonempty set of expressions. Unlike the unification algorithm for a one-sorted resolution, the WR-unification algorithm requires a finite set OA of order axioms as an input in addition to a finite nonempty set W of expressions. OA is needed in the algorithm since the UWR-resolution needs to know the unary predicate symbols determined by the outermost symbols of terms and variables. For example, if t_i and v_j are, respectively, a term and a variable that are to be unified, then the WR-unification algorithm needs to determine whether $Ran(t_i) \sqsubseteq Ran(v_j)$ or what are $IM(Ran(t_i))$ and $IM(Ran(v_j))$. These become determinable if OA is provided as an input to the WR-unification algorithm. The following process is applicable to a finite nonempty set W of expressions and a finite set OA of ordering axioms:

WR-Unification Algorithm

Step 1 Set $k = 0$, $W_k = W$, $\sigma_k = \epsilon$, and go to Step 2.

Step 2 If W_k is a singleton, stop; σ_k is a wr-mgu for W. Otherwise, find the disagreement set D_k of W_k and go to Step 3.

Step 3 If there exist elements v_k and t_k in D_k such that v_k is a variable that does not occur in t_k, go to Step 4. Otherwise, stop; W is not unifiable.

Step 4 If $Ran(t_k) \sqsubseteq Ran(v_k)$, then let $\sigma_{k+1} = \sigma_k\{t_k/v_k\}$, $W_{k+1} = W_k\{t_k/v_k\}$, and go to Step 7. Otherwise, go to Step 5.

Step 5 If t_k is a variable and $IM(Ran(v_k)) \cap IM(Ran(t_k)) \neq \phi$, then go to Step 6. Otherwise, stop; W is not unifiable.

Step 6 If $|\, IM(Ran(v_k)) \cap IM(Ran(t_k)) \,| = 1$, then let $\sigma_{k+1} = \sigma_k \{w/v_k , w/t_k\}$ where w is a variable satisfying $Ran(w) = IM(Ran(v_k)) \cap IM(Ran(t_k))$, $W_{k+1} = W_k \{ w/v_k , w/t_k \}$, and go to Step 7. Otherwise [i.e., $|\, IM(Ran(v_k)) \cap IM(Ran(t_k)) \,| > 1$], do the following: (i) let P be a new unary predicate that is not in OA ; (ii) for each $Q \in IM(Ran(v_k)) \cap IM(Ran(t_k))$, enter $Q \subseteq P$ in OA , and also enter $P \subseteq P_{v_k}$ and $P \subseteq P_{t_k}$ in OA where P_{v_k} and P_{t_k} are the predicates indicated by $Ran(v_k)$ and $Ran(t_k)$, respectively; and (iii) let $\sigma_{k+1} = \sigma_k \{ w/v_k , w/t_k \}$ where w is a variable satisfying $Ran(w) = P$, $W_{k+1} = W_k \{ w/v_k , w/t_k \}$ and go to Step 7.

Step 7 Set $k = k+1$ and go to Step 2.

There are two basic properties of the WR-unification algorithm that need to be justified. One is that the preceding process always terminates for any finite nonempty set of well-formed expressions. The other is that for a unifiable set of expressions, the outcome of a wr-mgu is always ensured. The former property can be shown in a straightforward way. The algorithm has three termination points at Steps 2, 3 and 5, respectively. If the algorithm does not terminate at any of these points and continue infinitely, the algorithm would generate an infinite sequence $W\sigma_0$, $W\sigma_1$, $W\sigma_2$, $\cdots$, of finite nonempty sets of expressions with the property that each successive set contains one less variable than its predecessor, namely, $W\sigma_k$ contains v_k but $W\sigma_{k+1}$ does not. This is impossible since W contains only finitely many distinct variables. Therefore the algorithm must terminate in a finite number of steps. The latter property, i.e. a wr-mgu is ensured for a nonempty finite unifiable set of expressions, is formally shown by the theorem given below. This result is used in the proof of Lemma 11.3.4 and elsewhere.

<u>Theorem 11.3.3</u> WR-Unification Theorem.

Given a finite set OA of ordering axioms and a finite nonempty unifiable set W of expressions, the WR-unification algorithm always terminates at step 2 and the last σ_k is a wr-mgu for W .

Proof. From the hypothesis that W is unifiable, there is a substitution θ that unifies W . It suffices to prove then that the WR-unification algorithm always terminates at Step 2; and that for each $k \geq 0$ until the WR-unification algorithm so terminates, $\theta = \sigma_k \lambda_k$ holds at Step 2 for some substitution λ_k [this is sufficient enough to say that the last σ_k is a wr-mgu for W because the last σ_k possibly includes one or more wr-subpairs that would be introduced at Step 6]. This is proved by induction on k .

For $k = 0$, by taking $\lambda_0 = \theta$, $\theta = \sigma_0 \lambda_0$ since $\sigma_0 = \epsilon$. For $0 \leq k \leq n$, assume that $\theta = \sigma_k \lambda_k$ holds at Step 2 for some substitution λ_k . When $k = n$ only two cases are possible at Step 2: either (i) $W\sigma_n$ is a singleton or (ii) $W\sigma_n$ is not a singleton. In case (i), the WR-unification algorithm terminates at Step 2 and σ_n is a wr-mgu since by the induction hypothesis $\theta = \sigma_n \lambda_n$ for some substitution λ_n . In case (ii), the WR-unification algorithm finds a disagreement set D_n of $W\sigma_n$. The inductive step is then to show that in case (ii) the process continues and does not terminate either at Step 3 or 5, and, when $k = n + 1$, $\theta = \sigma_{n+1}\lambda_{n+1}$ holds for some substitution λ_{n+1} .

Let $k = n$. It must hold that λ_n unifies D_n from the followings: θ is a unifier of W , $\theta = \sigma_n \lambda_n$ holds at Step 2 by the induction hypothesis, and D_n is the disagreement set of $W\sigma_n$. At Step 3, since W is unifiable, there must exist a variable v_n and a term t_n in D_n that is different from v_n . Here since λ_n unifies D_n , it holds that

$$v_n \lambda_n = t_n \lambda_n \quad \cdots \text{ (1)} .$$

From (1) it can be shown that v_n never occurs in t_n : If v_n occurs in t_n , then $v_n \lambda_n$ occurs in $t_n \lambda_n$. Since it is impossible that, while v_n and t_n are distinct, $v_n \lambda_n$ occurs in $t_n \lambda_n$ and at the same time $v_n \lambda_n = t_n \lambda_n$, v_n can not occur in t_n . Therefore the WR-unification algorithm will not terminate at Step 3, but will go Step 4.

At Step 4 the algorithm sets $\sigma_{n+1} = \sigma_n \{t_n / v_n\}$ if $Ran(t_n) \subseteq Ran(v_n)$ and otherwise will go to Step 5. At Step 5, as long as D_n is unifiable, it is neither possible that $Ran(t_n) \nsubseteq Ran(v_n)$ when t_k is not a variable nor $IM(Ran(t_n)) \cap IM(Ran(v_n)) = \phi$. This implies that the algorithm never terminates at Step 5, but will go Step 6. At Step 6, the algorithm sets either

$$\sigma_{n+1} = \sigma_n \{t_n / v_n\} \quad \cdots \text{ (2)}$$

$$\text{or} \qquad \sigma_{n+1} = \sigma_n \{w_n / v_n \ , \ w_n / t_n\} \quad \cdots \text{ (3)}$$

where w_n is a new variable that does not occur either in v_n or in t_n . Let case (a) be when the algorithm sets σ_{n+1} as (2) and let case (b) be when the algorithm sets σ_{n+1} as (3). In the rest of the proof, it is shown that at Step 2 in both cases (a) and (b) $\theta = \sigma_{n+1}\lambda_{n+1}$ holds for some substitution λ_{n+1} .

Case (a) : Let $\lambda_{n+1} = \lambda_n - \{(t_n \lambda_n)/v_n\}$. Then

$\lambda_n = \{(t_n \lambda_n)/v_n\} \cup \lambda_{n+1}$	by definition of λ_{n+1}
$= \{(t_n \lambda_{n+1})/v_n\} \cup \lambda_{n+1}$	since v_n does not occur in t_n
$= \{t_n/v_n\}\lambda_{n+1} \cdots (4)$	from the properties† of the composition of substitutions.

Therefore the following holds:

$\theta = \sigma_n \lambda_n$	from the induction hypothesis
$= \sigma_n \{t_n/v_n\}\lambda_{n+1}$	from (4)
$= \sigma_{n+1}\lambda_{n+1} \cdots (5)$	from (2) .

Case (b) : Let $\lambda_{n+1} = \lambda_n - \{(w\lambda_n)/v_n , (w\lambda_n)/t_n\}$ [notice here that t_n is a variable]. Then

$\lambda_n = \{(w\lambda_n)/v_n , (w\lambda_n)/t_n\} \cup \lambda_{n+1}$	by definition of λ_{n+1}
$= \{(w\lambda_{n+1})/v_n , (w\lambda_{n+1})/t_n\} \cup \lambda_{n+1}$	since w does not occur either in v_n or in t_n
$= \{w/v_n , w/t_n\}\lambda_{n+1} \cdots (6)$	from the properties of the composition of substitutions

Therefore the following holds:

$\theta = \sigma_n \lambda_n$	from the induction hypothesis
$= \sigma_n \{w/v_n , w/t_n\}\lambda_{n+1}$	from (6)
$= \sigma_{n+1}\lambda_{n+1} \cdots (7)$	from (3) .

Hence from (5) and (7), it follows that for all $k \geq 0$, $\theta = \sigma_k \lambda_k$ holds at Step 2 for some substitution λ_k . Since the WR-unification algorithm must terminate but it will never terminate either at Step 3 or 5, it must terminate at Step 2. Conse-

† Properties of the composition of substitutions include: (1) for any expression E and any substitutions σ and λ , $(E\sigma)\lambda = E(\sigma\lambda)$, (2) for any substitutions σ and λ , if $E\sigma = E\lambda$ then $\sigma = \lambda$, (3) for any substitutions σ , λ and δ , $(\sigma\lambda)\delta = \sigma(\lambda\delta)$, and (4) for any sets A and B of expressions and substitution λ , $(A \cup B)\lambda = A\lambda \cup B\lambda$.

quently, whenever the algorithm terminates at Step 2 the last σ_k is a wr-mgu for W . *Q.E.D.*

Now the UWR-resolution operator, denoted by $R_W(\cdot)$, is introduced in a similar way as the ground resolution operator $R(\cdot)$ was introduced at the beginning of this section. $R_W(\cdot)$ differs from $R(\cdot)$ only by the fact that the former is applied to the clauses in L_Σ^1, whereas the latter is applied to the ground clauses.

Definition 11.3.1

Given $< OA , T_\Sigma >$, *the UWR-resolution of* T_Σ, denoted by $R_W(T_\Sigma)$, is the set of all clauses consisting of members of T_Σ and all wr-resolvents of all pairs of members of T_Σ. *The* n^{th} *UWR-resolution of* T_Σ, denoted by $R_W^n(T_\Sigma)$, is then defined as follows: $R_W^0(T_\Sigma) = T_\Sigma$ and for $n \geq 0$, $R_W^{n+1}(T_\Sigma) = R_W(R_W^n(T_\Sigma))$. ●

Now a lemma called the Lifting Lemma is proved. Before doing so, the notion called *standardization* is introduced in order to make a pair of clauses not share any common variable. If L is a clause and $v_1 , \cdots , v_k$ are all the distinct variables, in alphabetical order, which occur in L , then the x-standardization of L , denoted by ξ_L , is the substitution $\{x_1/v_1 , \cdots , x_k/v_k\}$ where $Ran(x_i) = Ran(v_i)$, $1 \leq i \leq k$, and the y-standardization of L , denoted by η_L , is the substitution $\{y_1/v_1 , \cdots , y_k/v_k\}$ where $Ran(y_i) = Ran(v_i)$, $1 \leq i \leq k$. It is said that L is x-standardized (y-standardized) to $L\,\xi_L$ ($L\,\eta_L$). It is noticed that for a given pair of clauses, say C and D , $C\,\xi_C$ and $D\,\eta_D$ share no common variable.

Lemma 11.3.4 Lifting Lemma.

Given a $< OA \ , T_\Sigma >$, if P_Σ is any subset of the Herbrand universe of T_Σ , then $R(P_\Sigma(T_\Sigma)) \subseteq P_\Sigma(R_W(T_\Sigma))$.

Proof. Suppose $E \in R(P_\Sigma(T_\Sigma))$. Either $E \in P_\Sigma(T_\Sigma)$ or E is a ground resolvent of two ground clauses in $P_\Sigma(T_\Sigma)$. If $E \in P_\Sigma(T_\Sigma)$, then $E \in P_\Sigma(R_W(T_\Sigma))$ since $T_\Sigma \subseteq R_W(T_\Sigma)$. Therefore, it suffices to show that when E is a ground resolvent of two ground clauses in $P_\Sigma(T_\Sigma)$, then $E \in P_\Sigma(R_W(T_\Sigma))$. Let E be a ground resolvent of two ground clauses, say C^{gr} and D^{gr} . Since C^{gr}, $D^{gr} \in P_\Sigma(T_\Sigma)$, there are two clauses, say C and D , in T_Σ and two substitutions, say α and β , satisfying the following: If $\alpha = \{t_1/v_1, \cdots, t_k/v_k\}$ where $v_1, \cdots, v_k$ are all distinct variables in C and $\beta = \{w_1/u_1, \cdots, w_m/u_m\}$ where $u_1, \cdots, u_m$ are all distinct variables in D , then (i) $C^{gr} = C\alpha$ and $D^{gr} = D\beta$ and (ii) α and β are over† $P_\Sigma(T_\Sigma)$, i.e., $t_1, \cdots, t_k$ and $w_1, \cdots, w_m$ are in P_Σ . Then from the fact that E is a ground resolvent of C^{gr} and D^{gr} , it follows that

$$E = (C - L)\alpha \cup (D - M)\beta ,$$

where $L \subseteq C$ and $M \subseteq D$, L and M are nonempty, and $L\alpha$ and $M\beta$ are singletons whose respective members are complements [notice that if $L = C$ and $M = D$, then E is $\square$]. Let θ be

$$\theta = \{ t_1/x_1, \cdots, t_k/x_k, w_1/y_1, \cdots, w_m/y_m \} .$$

Then it follows that

$$E = (C - L)\xi_C\theta \cup (D - M)\eta_D\theta \cdots (1)$$

† If P is any set of terms and the terms $t_1, \cdots, t_n$ of the components of the substitution $\theta = \{t_1/v_1, \cdots, t_n/v_n\}$ are all in P , then θ is a substitution over P .

and that $L\xi_C\theta = L\alpha$ and $M\eta_D\theta = M\beta$. Let $N(L\xi_C \cup M\eta_D)$ stand for the set of atomic formulas that are members, or complements of members, of the set $L\xi_C \cup M\eta_D$ [this convention is used in the rest of Part II]. Then θ unifies $N(L\xi_C \cup M\eta_D)$. By the WR-unification theorem, there is a wr-mgu σ_W unifying $N(L\xi_C \cup M\eta_D)$ so that for some substitution λ,

$$\theta = \sigma_W\lambda \quad \cdots \text{ (2).}$$

Here λ is over P_Σ since the substitution θ is over P_Σ. It follows that $L\xi_C\sigma_W\lambda = L\alpha$ and $M\eta_D\sigma_W\lambda = M\beta$. Then since $L\alpha$ and $M\beta$ are singletons whose respective members are complements, so are $L\xi_C\sigma_W$ and $M\eta_D\sigma_W$. Let F be

$$F = (C - L)\xi_C\sigma_W \cup (D - M)\eta_D\sigma_W \quad \cdots \text{ (3).}$$

Then it follows that $F \in R_W(T_\Sigma)$ since (3) implies that F is a wr-resolvent of C and D. From (1), (2) and (3) it also follows that $E = F\lambda$†. Finally, since λ is over P_Σ, it is concluded that $E \in P_\Sigma(R_W(T_\Sigma))$. *Q.E.D.*

<u>Example 11.3.1</u>

Consider the following $< OA\ , T_\Sigma >$:

$$\begin{array}{lll} OA: & (1) & D \subseteq B\ ,\ D \subseteq C\ , \\ & (2) & E \subseteq B\ ,\ E \subseteq C\ , \\ T_\Sigma: & (3) & P(z^{\Sigma B}) \cup Q(z^{\Sigma B}\ , f^D(z^{\Sigma B}))\ , \\ & (4) & \neg P(z^{\Sigma C})\ . \end{array}$$

The Herbrand universe of T_Σ is the following:

$$\{d^D,\ e^E,\ f^D(d^D),\ f^D(e^E),\ f^D(f^D(d^D)),\ f^D(f^D(e^E)),\ \cdots\ ,\}\ .$$

† Distributive property holds for substitutions: $(A \cup B)\lambda = A\lambda \cup B\lambda$.

where d^D and e^E are the constants such that $Ran(d^D) = D$ and $Ran(e^E) = E$. Let a finite subset P_Σ of the Herbrand universe of T_Σ be $P_\Sigma = \{ e^E , d^D , f^D(e^E) , f^D(d^D) \}$. Then the saturation of T_Σ over P_Σ is

$$\begin{aligned} P_\Sigma(T_\Sigma) = \{ & P(e^E) \cup Q(e^E, f^D(e^E)) , P(d^D) \cup Q(d^D, f^D(d^D)) , \\ & P(f^D(e^E)) \cup Q(f^D(e^E), f^D(f^D(e^E))) , \\ & P(f^D(d^D)) \cup Q(f^D(d^D), f^D(f^D(d^D))) , \\ & \neg P(e^E) , \neg P(d^D) , \neg P(f^D(e^E)) , \neg P(f^D(d^D)) \} . \end{aligned}$$

$$\begin{aligned} R(P_\Sigma(T_\Sigma)) = P_\Sigma(T_\Sigma) \cup \{ & Q(e^E, f^D(e^E)) , Q(d^D, f^D(d^D)) , \\ & Q(f^D(e^E), f^D(f^D(e^E))) , Q(f^D(d^D), f^D(f^D(d^D))) \} . \end{aligned}$$

On the other hand, let a new predicate K be defined as $\forall x \ (K(x) \leftrightarrows B(x) \cap C(x))$, then

$$R_W(T_\Sigma) = \{ P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, f^D(x^{\Sigma B})) , \neg P(x^{\Sigma C}) , Q(x^{\Sigma K}, f^D(x^{\Sigma K})) \} .$$

$$\begin{aligned} P_\Sigma(R_W(T_\Sigma)) = P_\Sigma(T_\Sigma) \cup \{ & Q(e^E, f^D(e^E)) , Q(d^D, f^D(d^D)) , \\ & Q(f^D(e^E), f^D(f^D(e^E))) , Q(f^D(d^D), f^D(f^D(d^D))) \} . \end{aligned}$$

It clearly follows that $R(P_\Sigma(T_\Sigma)) \subseteq P_\Sigma(R_W(T_\Sigma))$.

The following is a corollary to the Lifting Lemma which shows that the n^{th} UWR-resolutions are also semicommutative with saturation.

Corollary 11.3.5

Given a $< OA , T_\Sigma >$, if P_Σ is any subset of the Herbrand universe of T_Σ , then $R^n(P_\Sigma(T_\Sigma)) \subseteq P_\Sigma(R_W^n(T_\Sigma))$.

Proof. Proof is by induction on n . For $n = 0$, $R^0(P_\Sigma(T_\Sigma)) = P_\Sigma(T_\Sigma) = P_\Sigma(R_W^0(T_\Sigma))$. For $n \geq 0$, let $R^n(P_\Sigma(T_\Sigma)) \subseteq P_\Sigma(R_W^n(T_\Sigma))$ be hold. Then the inductive step is the following:

$$\begin{aligned} R^{n+1}(P_\Sigma(T_\Sigma)) &= R(R^n(P_\Sigma(T_\Sigma))) && \text{by definition of } R^{n+1}, \\ &\subseteq R(P_\Sigma(R_W^n(T_\Sigma))) && \text{by the induction hypothesis}, \\ &\subseteq P_\Sigma(R_W(R_W^n(T_\Sigma))) && \text{by Lemma 11.3.4}, \\ &= P_\Sigma(R_W^{n+1}(T_\Sigma)) && \text{by definition of } R_W^{n+1}. \end{aligned}$$

Q.E.D.

Now the following form of lemma is concluded:

<u>Lemma 11.3.6</u>

If a $< OA \ , T_\Sigma >$ is unsatisfiable, then for some finite subset P_Σ of the Herbrand universe of T_Σ and some $n \geq 0$, $P_\Sigma(R_W^n(T_\Sigma))$ contains $\square$.

Proof. By Corollary 11.3.5, it is immediately obtained from Theorem 11.3.2. *Q.E.D.*

Finally, the final version of the Herbrand theorem for L_Σ^1 clauses is proved, which assures the completeness of UWR-resolution.

<u>Theorem 11.3.7</u> Completeness of UWR-Resolution.

A $< OA \ , T_\Sigma >$ is unsatisfiable if and only if $R_W^n(T_\Sigma)$ contains $\square$ for some $n \geq 0$.

Proof. The "only if" part is proved first: Lemma 11.3.6 is considered. Here mere replacement of variables by terms cannot produce $\square$ for a nonempty clause, i.e., $P_\Sigma(R_W^n(T_\Sigma))$ will contain $\square$ if and only if $R_W^n(T_\Sigma)$ contains $\square$. Therefore, by simply replacing $P_\Sigma(R_W^n(T_\Sigma))$ of Lemma 11.3.6 by $R_W^n(T_\Sigma)$, the "only if" part of the theorem is immediately obtained.

Now the "if" part is proved: Let $R_W^n(T_\Sigma)$ contain $\Box$ for some $n \leq 0$. Suppose $< OA , T_\Sigma >$ is satisfiable. Since any resolvent of two clauses is a logical consequence of the two clauses, any structure satisfying $< OA , T_\Sigma >$ should also satisfy $\Box$. The empty clause $\Box$ is never satisfiable by any structure. Hence $< OA , T_\Sigma >$ is unsatisfiable. *Q.E.D.*

CHAPTER XII

EFFICIENCY OF UWR-RESOLUTION

12.1. A Hypothetic Many-Sorted Resolution

In this chapter, the efficiency of UWR-resolution is discussed. Informally speaking, the efficiency of UWR-resolution is due to letting a wr-resolvent subsume a class of resolvents that would otherwise need to be generated. For example, when a pair of clauses satisfying a certain condition is resolved, if UWR-resolution is employed, only one resolvent is generated by using one or more wr-subpairs, whereas otherwise more than one resolvents must be generated. In the latter case, resolvents can be generated that only lead to dead ends.

In order to discuss of the efficiency of UWR-resolution in an organized way, the situation described informally in the preceding paragraph must be formalized. As a way of doing this, a hypothetic many-sorted resolution scheme, namely Σ-resolution, is introduced. Informally speaking, the Σ-resolution is a many-sorted resolution that is identical with the UWR-resolution except that in the former wr-subpair is no longer used, but a pair of wr-substitution components called Σ-subpair is used in place where a wr-subpair is to be used. Unlike wr-subpairs, Σ-subpairs can be introduced without extending the language L_Σ^1 that is currently being used. The formal notion of a Σ-subpair is introduced shortly. Although here the efficiency of the

UWR-resolution is discussed by comparing it with the efficiency of the Σ-resolution, similar comparison can be made with the many-sorted resolution scheme such as that of Walther's [Walt83]. Both Walther's scheme and the Σ-resolution have the problem of generating useless resolvents as the ones described in Section 8.1.

Formally speaking, the Σ-resolution differs from the UWR-resolution only by the following. Let two variables v_i and v_j satisfy the following conditions: (i) $Ran(v_i) \nsubseteq Ran(v_j)$ and $Ran(v_j) \nsubseteq Ran(v_i)$, and (ii) $|IM(Ran(v_i)) \cap IM(Ran(v_j))| > 1$. Then in the UWR-resolution the variables v_i and v_j are unified by a wr-subpair $\{w/v_i, w/v_j\}$ where w is a variable satisfying that $Ran(w)$ is a new predicate with which $|IM(Ran(v_i)) \cap IM(Ran(v_j))| = 1$ and $Ran(w) \in IM(Ran(v_i)) \cap IM(Ran(v_j))$. Compared to this, in the Σ-resolution the variables v_i and v_j are unified by a pair of wr-substitution components $\{w'/v_i, w'/v_j\}$ where w' is a variable satisfying $Ran(w') \in IM(Ran(v_i)) \cap IM(Ran(v_j))$. The pair of wr-substitution components $\{w'/v_i, w'/v_j\}$ is called a ***Σ-subpair***. It is said that $\{w'/v_i, w'/v_j\}$ is a Σ-subpair corresponding to the wr-subpair $\{w/v_i, w/v_j\}$. For such Σ-subpair $\{w'/v_i, w'/v_j\}$, let $Ran(w')$ stand for *the unification predicate of* $\{w'/v_i, w'/v_j\}$. Then it is easy to see that for the wr-subpair $\{w/v_i, w/v_j\}$ there are as many corresponding Σ-subpairs as $|IM(Ran(v_i)) \cap IM(Ran(v_j))|$ whose unification predicates differ from each other.

Once the notion of Σ-subpair which corresponds to that of wr-subpair is introduced, the notions Σ-substitution, Σ-unifier, Σ-unification and Σ-mgu of the Σ-resolution can also be introduced, repectively, in the same way as the notions wr-substitution, wr-unifier, wr-unification and wr-mgu of the UWR-resolution were introduced. Their formal definitions are omitted to avoid possible redundancy. The unification algorithm for the Σ-resolution, namely the Σ-unification algorithm, can also be introduced identically with the WR-unification algorithm except some modification of Step 6. Step 6 of the Σ-unification algorithm is:

Step 6 If $| IM(Ran(v_k)) \cap IM(Ran(t_k)) | = 1$, then let $\sigma_{k+1} = \sigma_k \{w/v_k , w/t_k\}$ where w is a variable satisfying $Ran(w) = IM(Ran(v_k)) \cap IM(Ran(t_k))$, $W_{k+1} = W_k \{w/v_k , w/t_k\}$, and go to Step 7. Otherwise [i.e., $| IM(Ran(v_k)) \cap IM(Ran(t_k)) | > 1$], let $\sigma_{k+1} = \sigma_k \{w/v_k , w/t_k\}$ where w is a variable satisfying $Ran(w) \in IM(Ran(v_k)) \cap IM(Ran(t_k))$, $W_{k+1} = W_k \{w/v_k , w/t_k\}$, and go to Step 7.

Accordingly, the basic properties of the the Σ-unification algorithm can be justified with a theorem, namely Σ-unification theorem, i.e., for any finite nonempty set of unifiable expressions the Σ-unification algorithm terminates and for a unifiable set of expressions a possible outcome of Σ-mgu is always ensured. Formal introduction of the Σ-unification theorem is omitted to avoid the possible redundancy. The completeness of the Σ-resolution, however, is shown indirectly in the following section where the WR-resolution is compared with the Σ-resolution in terms of efficiency.

In the rest of this section the Σ-resolution is discussed in more detail. First, it is shown, in the form of a lemma, how a wr-mgu and a Σ-mgu are related to each other. This lemma is used in the proof of a lemma in the following section.

The following notion is first introduced: A finite set θ of wr-substitution components is called *a variable-for-variable substitution* if for each wr-substitution component $v_\Sigma/v_W \in \theta$, both v_Σ and v_W are variables.

Lemma 12.1.1

Given two clauses C and D, let $N(L\,\xi_C \cup M\eta_D)$† be unifiable where $L \subseteq C$ and $M \subseteq D$, and L and M are nonempty. If σ_W and σ_Σ^k are, respectively, a wr-mgu and a Σ-mgu, each of which unifies $N(L\,\xi_C \cup M\eta_D)$, then there is a variable-for-variable substitution θ satisfying

(i) for each $v_\Sigma/v_W \in \theta$, $Ran(v_\Sigma) \subseteq Ran(v_W)$, and

(ii) $\sigma_W\theta = \sigma_\Sigma^k$.

Proof. Let E_W and E_Σ be the wr-resolvent and the Σ-resolvent of C and D which are generated by using σ_W as a wr-mgu and σ_Σ^k as a Σ-mgu, respectively, i.e.,

$$E_W = (C - L)\xi_C\sigma_W \cup (D - M)\eta_D\sigma_W\ ,$$

$$E_\Sigma = (C - L)\xi_C\sigma_\Sigma^k \cup (D - M)\eta_D\sigma_\Sigma^k\ .$$

It is noticed that once C and D are x-standardized and y-standardized to $C\,\xi_C$ and $D\,\eta_D$, respectively, then $x_1, \cdots, x_k$ and $y_1, \cdots, y_m$ are all the distinct variables in $C\,\xi_C$ and $D\,\eta_D$, respectively. Let σ_W and σ_Σ^k be, respectively,

$$\sigma_W = \{\, t_1/x_1,\ \cdots,\ t_k/x_k\ ,\ w_1/y_1,\ \cdots,\ w_m/y_m \,\}\ ,$$

$$\sigma_\Sigma^k = \{\, u_1/x_1,\ \cdots,\ u_k/x_k\ ,\ v_1/y_1,\ \cdots,\ v_m/y_m \,\}\ .$$

† Previously in Section 11.3 $N(L\,\xi_C \cup M\eta_D)$ has been defined as the set of atomic formulas that are members, or complements of members, of the set $L\,\xi_C \cup M\eta_D$.

If some two-substitution components t_i/x_i , $w_j/y_j \in \sigma_W$, where $t_i = w_j$, constitute a wr-subpair $\{t_i/x_i , w_j/y_j\}$, then there is a Σ-subpair $\{u_i/x_i , v_j/y_j\}$, where u_i/x_i , $v_j/y_j \in \sigma_\Sigma^k$ and $u_i = v_j$, which corresponds to the wr-subpair $\{t_i/x_i , w_j/y_j\}$. From the way that a wr-subpair and a Σ-subpair are defined, the following relationship holds between $\{t_i/x_i , w_j/y_j\}$ and $\{u_i/x_i , v_j/y_j\}$:

$$\{t_i/x_i , w_j/y_j\}\lambda = \{u_i/x_i , v_j/y_j\} ,$$

where $\lambda = \{u_i/t_i\}$ and $Ran(u_i) \subseteq Ran(t_i)$. Now let a substitution θ be constructed in the following way: (i) if $\{t_i/x_i , w_j/y_j\} \subseteq \sigma_W$ is a wr-subpair and $\{u_i/x_i , v_j/y_j\} \subseteq \sigma_\Sigma^k$ is its corresponding Σ-sub pair, then $\{u_i/t_i\}$ is an element of θ , and (ii) no other wr-substitution components than those identified by (i) are the elements of θ . Then it follows that

$$\sigma_W \theta = \sigma_\Sigma^k$$

and for any substitution component $v_\Sigma/v_W \in \theta$, $Ran(v_\Sigma) \subseteq Ran(v_W)$. *Q.E.D.*

Now, the Σ-resolution operator $R_\Sigma(\cdot)$ is introduced in a way similar to that for the UWR-resolution operator $R_W(\cdot)$ was introduced:

Definition 12.1.1

Given a $< OA , T_\Sigma >$, *the Σ-resolution of* T_Σ, denoted by $R_\Sigma(T_\Sigma)$, is the set of all clauses consisting of members of T_Σ and all Σ-resolvents of all pairs of members of T_Σ. *The* n^{th} *Σ-resolution of* T_Σ, denoted by $R_\Sigma^n(T_\Sigma)$, is then defined as follows: $R_\Sigma^0(T_\Sigma) = T_\Sigma$, and for $n \geq 0$, $R_\Sigma^{n+1}(T_\Sigma) = R_\Sigma(R_\Sigma^n(T_\Sigma))$. •

In the following it is illustrated how $R_W(\cdot)$ and $R_\Sigma(\cdot)$ are carried out for a given set of L^1_Σ clauses. Here the result of Lemma 12.1.1. is also illustrated.

Example 12.1.1

Let the $< OA , T_\Sigma >$ of Example 10.1.2 be augmented as follows:

OA †: (1) $D \subseteq B$, $D \subseteq C$,

(2) $E \subseteq B$, $E \subseteq C$,

(3) $F \subseteq D$, $F \subseteq E$,

(4) $G \subseteq D$, $G \subseteq E$,

(5) $H \subseteq D$, $H \subseteq E$,

T_Σ : (6) $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, y^{\Sigma E}) \cup R(f^F(x^{\Sigma B}), y^{\Sigma E})$,

(7) $\neg P(x^{\Sigma C})$,

(8) $\neg Q(x^{\Sigma E}, f^H(x^{\Sigma E}))$,

(9) $\neg R(x^{\Sigma E}, x^{\Sigma D})$.

(i) $R_W(\cdot)$ is performed as follows:

$$R^1_W(T_\Sigma) = R^0_W(T_\Sigma) \cup \{ Q(x^{\Sigma K}, y^{\Sigma E}) \cup R(f^F(x^{\Sigma K}), y^{\Sigma E}),\ P(x^{\Sigma E}) \cup R(f^F(x^{\Sigma E}), f^H(x^{\Sigma E})),\ P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma J}) \},$$

where two new predicates K and J are defined by $\forall x\ (K(x) \leftrightarrows B(x) \cap C(x))$ and $\forall x\ (J(x) \leftrightarrows D(x) \cap E(x))$:

$$R^2_W(T_\Sigma) = R^1_W(T_\Sigma) \cup \{ R(f^F(z^{\Sigma E}), f^H(z^{\Sigma E})),\ Q(z^{\Sigma K}, z^{\Sigma J}),\ \cdots \}.$$

Here $Q(x^{\Sigma K}, x^{\Sigma J})$ is a wr-resolvent of $\neg P(y^{\Sigma C})$, $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma J}) \in R^1_W(T_\Sigma)$.

† For simplicity, the ordering axioms that are derivable from OA , such as $F \subseteq B$, $G \subseteq B$, $H \subseteq B$, $F \subseteq C$, $G \subseteq C$ and $H \subseteq C$, are omitted in OA . This convention is used in the rest of Part II.

(ii) $R_\Sigma(\cdot)$ is carried out as follows:

$$\begin{aligned} R_\Sigma^1(T_\Sigma) = R_\Sigma^0(T_\Sigma) \cup \{ & Q(x^{\Sigma D}, y^{\Sigma E}) \cup R(f^F(x^{\Sigma D}), y^{\Sigma E}), \\ & Q(x^{\Sigma E}, y^{\Sigma E}) \cup R(f^F(x^{\Sigma E}), y^{\Sigma E}), \\ & P(x^{\Sigma E}) \cup R(f^F(x^{\Sigma E}), f^H(x^{\Sigma E})), \\ & P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma F}), \\ & P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma G}), \\ & P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma H})\}. \end{aligned}$$

$$\begin{aligned} R_\Sigma^2(T_\Sigma) = R_\Sigma^1(T_\Sigma) \cup \{ & R(f^F(w^{\Sigma E}), f^H(w^{\Sigma E})),\ Q(w^{\Sigma D}, w^{\Sigma F}), \\ & Q(w^{\Sigma E}, w^{\Sigma F}),\ Q(w^{\Sigma D}, w^{\Sigma G}),\ Q(w^{\Sigma E}, w^{\Sigma G}), \\ & Q(w^{\Sigma D}, w^{\Sigma H}),\ Q(w^{\Sigma E}, w^{\Sigma H}), \\ & R(f^F(w^{\Sigma F}), f^H(w^{\Sigma F})),\ R(f^F(w^{\Sigma G}), f^H(w^{\Sigma G})), \\ & R(f^F(w^{\Sigma H}), f^H(w^{\Sigma H})),\ \cdots \}. \end{aligned}$$

Now consider the wr-resolvent $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma I}) \in R_W^1(T_\Sigma)$ and the Σ-resolvent $P(y^{\Sigma B}) \cup Q(y^{\Sigma B}, y^{\Sigma F}) \in R_\Sigma^1(T_\Sigma)$ each of which is obtained by resolving (6) and (9). The wr-resolvent is obtained by using the wr-mgu $\{f^F(z^{\Sigma B})/x^{\Sigma E}, x^{\Sigma I}/x^{\Sigma D}, x^{\Sigma I}/y^{\Sigma E}\}$, say σ_W, and the Σ-resolvent, by using the Σ-mgu $\{f^F(x^{\Sigma B})/x^{\Sigma E}, x^{\Sigma F}/x^{\Sigma D}, x^{\Sigma F}/y^{\Sigma E}\}$, say σ_Σ^F. Here $\{x^{\Sigma I}/x^{\Sigma D}, x^{\Sigma I}/y^{\Sigma E}\}$ is a wr-subpair in the wr-mgu σ_W and $\{x^{\Sigma F}/x^{\Sigma D}, x^{\Sigma F}/y^{\Sigma E}\}$ is its corresponding Σ-subpair in the Σ-mgu σ_Σ^F. It follows that there is a variable-for-variable substitution $\{x^{\Sigma F}/x^{\Sigma I}\}$ satisfying

$$\sigma_W\{x^{\Sigma F}/x^{\Sigma I}\} = \sigma_\Sigma^F$$

where $Ran(x^{\Sigma F}) \Subset Ran(x^{\Sigma I})$. The additionally generated $R_W^2(T_\Sigma)$ and $R_\Sigma^2(T_\Sigma)$ will be used later in Example 12.2.1 and in Example 12.2.2.

12.2. UWR-Resolution vs Hypothetic Many-Sorted Resolution

In this section the efficiency of UWR-resolution is discussed. The efficiency is discussed by comparing UWR-resolution with Σ-resolution in a certain way, i.e., for a given many-sorted theory $< OA \, , T_\Sigma >$, by using $R_W(\cdot)$ and $R_\Sigma(\cdot)$ two refutations of the theory are derived. Both refutations are generated by the method called *level-saturation,* i.e., the resolution operators $R_W(\cdot)$ and $R_\Sigma(\cdot)$ are consecutively applied to the results of the application at the previous level until $\Box$ is derived. Each refutation is then the alignment of all the resolvents ordered according to their generations.

The way the two resolution schemes are compared here is, in fact, to contrast the longest possible refutation sequences that can be generated for a given theory by the two schemes $R_W(\cdot)$ and $R_\Sigma(\cdot)$. This approach is meaningful in the sense that at each level it is not known a priori to the resolution which two clauses should be best resolved. The worst case may result in generating all the possible resolvents at each level. The approach adopted here consists of two stages. The first stage is to show that given a many-sorted theory $< OA \, , T_\Sigma >$ the length of the shortest (by level) refutation generated by $R_W(\cdot)$ is identical with that generated by $R_\Sigma(\cdot)$. Then the second stage is to show that the total number of wr-resolvents generated by $R_W(\cdot)$ is smaller or equal to that generated by $R_\Sigma(\cdot)$. When the results from the two stages are combined, the intended comparison of the two resolution scheme is obtained.

In the following, the first stage is introduced. First, the notion "subsume" is introduced: Let C_1 and C_2 be two clauses that differ only by their variables. Let $\{v_1 , \cdots , v_n\}$ and $\{u_1 , \cdots , u_n\}$ be all the distinct variables in C_1 and in

C_2, respectively, where v_i in C_1 corresponds to u_i in C_2 for $1 \leq i \leq n$. When it is convenient, the notation $C_2 | v_k$ is often used to mean the v_k's corresponding variable in C_2, i.e., u_k. C_1 *subsumes* C_2 if for any $i \in \{1, \cdots, n\}$ $Ran(u_i) \subseteq Ran(v_i)$.

Lemma 12.2.1

Given a $< OA, T_\Sigma >$, if there is a clause $E_1 \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$, $i \geq 0$, then there is a clause $E_2 \in R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$ that subsumes E_1.

Proof. Poof is by induction on i. For $i = 0$, the proof is trivial since $R_W^0(T_\Sigma) = R_\Sigma^0(T_\Sigma)$. For $i > 0$, it is assumed that the following holds: For any clause $E_1 \in R_\Sigma^i(T_\Sigma) - R_\Sigma^{i-1}(T_\Sigma)$, there is a clause $E_2 \in R_W^i(T_\Sigma) - R_W^{i-1}(T_\Sigma)$ which subsumes E_1. The inductive step is then the following: Let $E_1 \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$ be a resolvent of two clauses C_1, $D_1 \in R_\Sigma^i(T_\Sigma)$. Let the abbreviated notations ξ_i and η_j be used for ξ_{C_i} and η_{D_j}, respectively [this way of abbreviating the $x(y)$-standardization is used in the rest of this section]. Then there are nonempty subsets $L_1 \subseteq C_1$, $M_1 \subseteq D_1$ such that

$$E_1 = (C_1 - L_1)\xi_1\sigma_\Sigma \cup (D_1 - M_1)\eta_1\sigma_\Sigma \cdots (1),$$

where σ_Σ unifies $N(L_1\xi_1 \cup M_1\eta_1)$. By the induction hypothesis, there are two clauses C_2, $D_2 \in R_W^i(T_\Sigma)$ which subsume C_1 and D_1, respectively. Let $\{x_1^1, \cdots, x_n^1\}$ and $\{x_1^2, \cdots, x_n^2\}$ be all the distinct variables in $C_1\xi_1$ and $C_2\xi_2$, respectively, and let $\{y_1^1, \cdots, y_m^1\}$ and $\{y_1^2, \cdots, y_m^2\}$ be the distinct variables in $D_1\eta_1$ and $D_2\eta_2$, respectively. Let δ_C and δ_D be

$$\delta_C = \{ x_1^1/x_1^2 , \cdots , x_n^1/x_n^2 \} ,$$

$$\delta_D = \{ y_1^1/y_1^2 , \cdots , y_m^1/y_m^2 \}.$$

Then since C_2 and D_2 subsume C_1 and D_1, respectively, it holds that $C_1\xi_1 = C_2\xi_2\delta_C$ and $D_1\eta_1 = D_2\eta_2\delta_D$, for any $i \in \{ 1, \cdots, n \}$, $Ran(x_i^1) \subseteq Ran(x_i^2)$ and for any $j \in \{ 1, \cdots, m \}$, $Ran(y_j^1) \subseteq Ran(y_j^2)$. It also holds that there are two singleton subsets $L_2 \subseteq C_2$ and $M_2 \subseteq D_2$ which are subsumed by L_1 and M_1, respectively. It follows that $L_1\xi_1 = L_2\xi_2\delta_C$ and $M_1\xi_1 = M_2\xi_2\delta_D$. Therefore, from (1) it follows that

$$E_1 = (C_2 - L_2)\xi_2\delta_C\sigma_\Sigma \cup (D_2 - M_2)\eta_2\delta_D\sigma_\Sigma \quad \cdots \quad (2).$$

Let δ be $\delta = \{x_1^1/x_1^2 , \cdots , x_n^1/x_n^2 , y_1^1/y_1^2 , \cdots , y_m^1/y_m^2\}$. Then it follows that

$$E_1 = ((C_2 - L_2)\xi_2 \cup (D_2 - M_2)\eta_2)\delta\sigma_\Sigma \quad \cdots \quad (3).$$

It also holds that

$$\begin{aligned} N(L_1\xi_1 \cup M_1\eta_1) &= N(L_2\xi_2\delta_C \cup M_2\eta_2\delta_D) \\ &= N((L_2\xi_2 \cup M_2\eta_2)\delta) . \end{aligned}$$

Therefore, when $N((L_1\xi_1 \cup M_1\eta_1)\delta)$ is unifiable by σ_Σ, $N(L_2\xi_2 \cup M_2\eta_2)$ is unifiable by $\delta\sigma_\Sigma$. Here $\delta\sigma_\Sigma$ is still a Σ-mgu for $N(L_2\xi_2 \cup M_2\eta_2)$ since δ simply replaces variables for variables and for each $v_1/v_2 \in \delta$ $Ran(v_1) \subseteq Ran(v_2)$. Now by the WR-unification theorem, it follows that there is a wr-mgu, say σ_W, unifying $N(L_2\xi_2 \cup M_2\eta_2)$. Then between σ_W and $\delta\sigma_\Sigma$, by Lemma 12.1.1, there exist a variable-for-variable substitution θ such that

$$\sigma_W\theta = \delta\sigma_\Sigma \quad \cdots \quad (4)$$

where for each substitution component $v_\Sigma/v_W \in \theta$, $Ran(v_\Sigma) \subseteq Ran(v_W)$. Let E_2 be the wr-resolvent of C_2 and D_2 that is generated by using σ_W as the wr-mgu, i.e.,

$$E_2 = (C_2 - L_2)\xi_2\sigma_W \cup (D_2 - M_2)\eta_2\sigma_W \quad \cdots \ (5) .$$

Then $E_2 \in R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$. Finally, the following holds:

$$\begin{aligned} E_1 &= ((C_2 - L_2)\xi_2 \cup (D_2 - M_2)\eta_2)\ \delta\ \sigma_\Sigma && \text{from (3)} \\ &= ((C_2 - L_2)\xi_2 \cup (D_2 - M_2)\eta_2)\ \sigma_W\ \theta && \text{from (4)} \\ &= E_2\theta && \text{from (5)} . \end{aligned}$$

Since for any substitution component $v_\Sigma/v_W \in \theta$, $Ran(v_\Sigma) \subseteq Ran(v_W)$. So it follows that there is $E_2 \in R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$ which subsumes E_1 . *Q.E.D.*

Example 12.2.1

Consider Example 12.1.1. Let $E_1 \in R_\Sigma^2(T_\Sigma) - R_\Sigma^1(T_\Sigma)$ be

$$E_1 = R(f^F(w^{\Sigma F}), f^H(w^{\Sigma F})) .$$

Here E_1 is a Σ-resolvent of $\neg\, Q(z^{\Sigma E}, f^H(z^{\Sigma E})) \in R_\Sigma^1(T_\Sigma)$, say C_1 , and $Q(z^{\Sigma D}, y^{\Sigma E}) \cup R(f^F(z^{\Sigma D}), y^{\Sigma E}) \in R_\Sigma^1(T_\Sigma)$, say D_1 . There are two clauses $\neg\, Q(z^{\Sigma E}, f^H(z^{\Sigma E}))$, $Q(z^{\Sigma K}, y^{\Sigma E}) \cup R(f^F(z^{\Sigma K}), y^{\Sigma E}) \in R_W^1(T_\Sigma)$, say C_2 and D_2 , respectively, which subsume C_1 and D_1 , respectively. A wr-resolvent of C_2 and D_2 , say E_2 , is then

$$E_2 = R(f^F(z^{\Sigma E}), f^H(z^{\Sigma E})) .$$

E_2 subsumes E_1 . It can be verified from Example 12.1.1 that $E_2 \in R_W^2(T_\Sigma) - R_W^1(T_\Sigma)$.

The following is a corollary to the preceding theorem which assures that the shortest deduction sequence generating $\square$ by $R_W(\cdot)$ is not longer than that by $R_\Sigma(\cdot)$.

Corollary 12.2.2

Given a $< OA , T_\Sigma >$, if n is the smallest non-negative integer for which $R^n_\Sigma(T_\Sigma)$ contains $\Box$, then $R^n_W(T_\Sigma)$ also contains $\Box$.

Proof. Let n be the smallest non-negative integer for which $R^n_\Sigma(T_\Sigma)$ contains $\Box$. Then $\Box$ is a resolvent of two clauses, say $C_1, D_1 \in R^{n-1}_\Sigma(T_\Sigma)$. Since $\Box$ is a resolvent of C_1 and D_1, the following holds: C_1 and D_1 are singletons, and there is a Σ-mgu, say σ_Σ, unifying $N(C_1\xi_1 \cup D_1\eta_1)$. It follows that

$$C_1\xi_1\sigma_\Sigma \cup D_1\eta_1\sigma_\Sigma = \Box$$

where $C_1\xi_1\sigma_\Sigma$ and $D_1\eta_1\sigma_\Sigma$ are singletons whose respective members are complements. By Lemma 12.2.1, there are two clauses in $R^{n-1}_W(T_\Sigma)$, say C_2 and D_2, which subsume C_1 and D_1, respectively. Here C_2 and D_2 are also singletons. It can be shown that $N(C_2\xi_2 \cup D_2\eta_2)$ is unifiable in a way similar to the one that showed $N(L_2\xi_2 \cup M_2\eta_2)$ was unifiable in the proof of Lemma 12.2.1. Let σ_W be a wr-mgu unifying $N(C_2\xi_2 \cup D_2\eta_2)$. Then $C_2\xi_2\sigma_W$ and $D_2\eta_2\sigma_W$ are singletons whose respective members are complements. It immediately follows that

$$C_2\xi_2\sigma_W \cup D_2\eta_2\sigma_W = \Box .$$

$R^n_W(T_\Sigma) - R^{n-1}_W(T_\Sigma)$ contains $\Box$. So does $R^n_W(T_\Sigma)$. *Q.E.D.*

Now the result in other direction to the result of Corollary 12.2.2 is derived. First, a few more notions associated with "subsume" are introduced. A set of clauses, denoted by $SBSM(C_1(v_k))$, $1 \leq k \leq n$, ***is subsumed by*** C_1 ***over*** $Ran(v_k)$ if for each $S \in IM(Ran(v_k))$ there is a clause $C_S \in SBSM(C_1(v_k))$ satisfying (i) C_1

subsumes C_S, (ii) $Ran(C_s \mid v_k) = S$, and (iii) $Ran(C_s \mid v_i) = Ran(v_i)$, if $i \neq k$. This notion is further generalized as follows. Let $v_1, \cdots, v_n$ be the variables in alphabetical order in C_1 and let $\{v_{i_1}, \cdots, v_{i_k}\} \subseteq \{v_1, \cdots, v_n\}$. Let Γ be the index set of $IM(Ran(v_{i_1})) \times \cdots \times IM(Ran(v_{i_k}))$. A set of clauses, denoted by $SBSM(C_1(v_{i_1}, \cdots, v_{i_k}))$, $\{v_{i_1}, \cdots, v_{i_k}\} \subseteq \{v_1, \cdots, v_n\}$, *is subsumed by* C_1 *over* $Ran(v_{i_1}), \cdots, Ran(v_{i_k})$, if the following is satisfied: For each $<S^l_{i_1}, \cdots, S^l_{i_k}> \in IM(Ran(v_{i_1})) \times \cdots \times IM(Ran(v_{i_k}))$, $l \in \Gamma$, there is a clause $C_S \in SBSM(C_1(v_{i_1}, \cdots, v_{i_k}))$ satisfying (i) C_1 subsumes C_S, (ii) for each v_h, $v_h \in \{v_{i_1}, \cdots, v_{i_k}\}$, $Ran(C_s \mid v_h) = S^l_h$, and (iii) for each $v_{h'}$, $v_{h'} \in \{v_1, \cdots, v_n\} - \{v_{i_1}, \cdots, v_{i_k}\}$, $Ran(C_s \mid v_i) = Ran(v_i)$.

In the rest of this section a symbol $<>$ is used to designate that $C_1 <> C_2$ means C_1 and C_2 subsume each other. The following notation is also used: Given a $< OA, T_\Sigma >$, let $\Delta^i(OA)$ stand for the set of all the unary predicate symbols that were newly defined from the 0^{th} UWR-resolution $R^0_W(T_\Sigma)$ to the i^{th} UWR-resolution $R^i_W(T_\Sigma)$. Then a variable v in a clause $C \in R^i_W(T_\Sigma)$ is a $d(i,j)$-*variable* if $Ran(v) \in \Delta^i(OA) - \Delta^j(OA)$.

<u>Lemma 12.2.3</u>

Given a $< OA, T_\Sigma >$, let there be a clause $E_1 \in R^{i+1}_W(T_\Sigma) - R^i_W(T_\Sigma)$. If there is no $d(i+1,0)$-variable in E_1, then there is a clause $E_2 \in R^{i+1}_\Sigma(T_\Sigma) - R^i_\Sigma(T_\Sigma)$ such that $E_1 <> E_2$. If there are some $d(i+1,0)$-variables, say $e_1, \cdots, e_k$, in E_1, then there is a $SBSM(E_1(e_1, \cdots, e_k)) \subseteq R^{i+1}_\Sigma(T_\Sigma) - R^i_\Sigma(T_\Sigma)$.

Proof. Proof is by induction on i. For $i = 0$, let a clause

$E_1 \in R_W^1(T_\Sigma) - R_W^0(T_\Sigma)$ be a resolvent of two clauses $C_1, D_1 \in R_W^0(T_\Sigma)$. Then it follows that for some two singleton subsets $L_1 \subseteq C_1$ and $M_1 \subseteq D_1$,

$$E_1 = (C_1 - L_1)\xi_1\sigma_W \cup (D_1 - M_1)\eta_1\sigma_W \quad \cdots \; (1),$$

where σ_W is a wr-mgu unifying $N(L_1\xi_1 \cup M_1\eta_1)$.

(Case I) If there is no $d(1,0)$-variable in E_1, then σ_W does not contain any wr-subpairs. Hence σ_W is also a Σ-mgu unifying $N(L_1\xi_1 \cup M_1\eta_1)$. This implies $E_1 \in R_\Sigma^1(T_\Sigma) - R_\Sigma^0(T_\Sigma)$ since $R_W^0(T_\Sigma) = R_\Sigma^0(T_\Sigma)$. It trivially holds $E_1 <> E_1$. Consequently, by letting $E_2 = E_1$, there is a $E_2 \in R_\Sigma^1(T_\Sigma) - R_\Sigma^0(T_\Sigma)$ such that $E_2 <> E_1$.

(Case II) If there are k $d(1,0)$-variables $e_1, \cdots, e_k$ in E_1, then the wr-mgu σ_W in (1) must contain k wr-subpairs since $E_1 \in R_W^1(T_\Sigma) - R_W^0(T_\Sigma)$. Let $\{x_1^1, \cdots, x_n^1\}$ and $\{y_1^1, \cdots, y_m^1\}$ be all the distinct variables in $C_1\xi_1$ and $D_1\eta_1$, respectively. Let the k wr-subpairs in σ_W be $\{e_1/x_{c(1)}^1, e_1/y_{d(1)}^1\}, \cdots, \{e_k/x_{c(k)}^1, e_k/y_{d(k)}^1\}$, where $\{x_{c(1)}^1, \cdots, x_{c(k)}^1\} \subseteq \{x_1^1, \cdots, x_n^1\}$, and $\{y_{d(1)}^1, \cdots, y_{d(k)}^1\} \subseteq \{y_1^1, \cdots, y_m^1\}$. Let Γ_e be the index set of $IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$. For each $\{S_1^l, \cdots, S_k^l\} \in IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$, $l \in \Gamma_e$, let λ^l be a substitution of k substitution components $\{v_1^l/e_1, \cdots, v_k^l/e_k\}$ such that $Ran(v_j^l) = S_j^l$, $1 \le j \le k$. Then when σ_W unifies $N(L_1\xi_1 \cup M_1\eta_1)$, $\sigma_W\lambda^l$ also unifies $N(L_1\xi_1 \cup M_1\eta_1)$ since λ^l simply substitutes variables $e_1, \cdots, e_k$ by $v_1^l, \cdots, v_k^l$, respectively. Here $\sigma_W\lambda^l$ is no longer a wr-mgu but a Σ-mgu since each wr-subpair $\{e_j/x_{c(j)}^1, e_j/y_{d(j)}^1\}$, $1 \le j \le k$, in σ_W is replaced by $\{v_j^l/x_{c(j)}^1, v_j^l/y_{d(j)}^1\}$ in $\sigma_W\lambda^l$ which is now a Σ-subpair. Let E_2^l be

$$E_2^l = (C_1 - L_1)\xi_1\sigma_W\lambda^l \cup (D_1 - M_1)\eta_1\sigma_W\lambda^l .$$

It is clear that $E_2^l \in R_\Sigma^1(T_\Sigma) - R_\Sigma^0(T_\Sigma)$ since $R_W^0(T_\Sigma) = R_\Sigma^0(T_\Sigma)$ and $\sigma_W\lambda^l$ is a Σ-mgu of $C_1, D_1 \in R_W^0(T_\Sigma)$. The following relationship holds between E_2^l and E_1:

$$\begin{aligned} E_2^l &= ((C_1 - L_1)\xi_1\sigma_W \cup (D_1 - M_1)\eta_1\sigma_W)\lambda^l \\ &= E_1\lambda^l . \end{aligned}$$

It follows that E_1 subsumes E_2^l since for any substitution component $v_\Sigma / v_W \in \lambda^l$ it holds that $Ran(v_\Sigma) \Subset Ran(v_W)$. Finally, the following is concluded: for each $\{S_1^l, \cdots, S_k^l\} \in IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$, $l \in \Gamma_e$, there is a clause $E_2^l \in R_\Sigma^1(T_\Sigma) - R_\Sigma^0(T_\Sigma)$ such that (i) E_1 subsumes E_2^l, (ii) $Ran(E_2^l \mid e_j) = S_j^l$, $1 \leq j \leq k$, and (iii) for any variable v in E_1 other than $e_1, \cdots, e_k$, $Ran(E_2^l \mid v) = Ran(v)$. Therefore, it follows that there is a $SBSM(E_1(e_1, \cdots, e_k)) \subseteq R_\Sigma^1(T_\Sigma) - R_\Sigma^0(T_\Sigma)$.

It is now assumed that for $i \geq 0$, the induction hypothesis holds. In the rest of the proof, the inductive step is shown.

Let $E_1 \in R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$ be a resolvent of two clauses $C_1, D_1 \in R_W^i(T_\Sigma)$. There are only three possible cases: (i) there is no $d(i,0)$-variable in either C_1 or D_1, (ii) some $d(i,0)$-variables are in either C_1 or D_1, and (iii) some $d(i,0)$-variables are in both C_1 and D_1. The inductive step for case (i) is similar to what was shown in the induction basis. The inductive step for case (iii) includes that for case (ii). Therefore in the rest of the proof it is only shown that the inductive step holds for case (iii).

First, some preliminary steps are given which are needed in the rest of the proof. From the fact that E_1 is a wr-resolvent of C_1 and D_1, let E_1 be

$$E_1 = (C_1 - L_1)\xi_1\sigma_W \cup (D_1 - M_1)\eta_1\sigma_W \ , \quad \cdots \ (2)$$

where $L_1 \subseteq C_1$ and $M_1 \subseteq D_1$, L_1 and M_1 are singletons, and σ_W is a wr-mgu unifying $N(L_1\xi_1 \cup M_1\eta_1)$. Here let $x_1^1, \cdots, x_n^1$ and $y_1^1, \cdots, y_m^1$ be all the distinct variables in $C_1\xi_1$ and $D_1\eta_1$, respectively. Let $x_{i_1}^1, \cdots, x_{i_c}^1$, $\{i_1, \cdots, i_c\} \subseteq \{1, \cdots, n\}$, and $y_{i_1}^1, \cdots, y_{i_d}^1$, $\{i_1, \cdots, i_d\} \subseteq \{1, \cdots, m\}$, be the $d(i,0)$-variables in $C_1\xi_1$ and $D_1\eta_1$, respectively. By the inductive hypothesis, there are $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$, $SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1)) \subseteq R_\Sigma^i(T_\Sigma)$.

By definition of $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$, it holds that for each clause, say α_C, in $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$ there is a substitution, say δ_C, such that $\alpha_C = C_1\xi_1\delta_C$. Hence let Δ_C be a set of substitutions such as

$$\Delta_C \stackrel{d}{=} \{\delta_C : C_1\xi_1\delta_C \in SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))\} .$$

An observation is made on Δ_C as follows: Let Γ_C be the index set of $IM(Ran(x_{i_1}^1)) \times \cdots \times IM(Ran(x_{i_c}^1))$. By definition of $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$, it follows that for each $l \in \Gamma_C$, if $<S_{i_1}^l, \cdots, S_{i_c}^l> \in IM(Ran(x_{i_1}^1)) \times \cdots \times IM(Ran(x_{i_c}^1))$, then there is the corresponding substitution $\delta_C^l \in \Delta_C$ such that $C_1\xi_1\delta_C^l \in SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$ and for each k, $i_1 \leq k \leq i_c$, $Ran(C_1\xi_1\delta_C^l \mid x_k^1) = S_k^l$.

Similarly, let Δ_D be defined from $SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1))$ as follows:

$$\Delta_D \stackrel{d}{=} \{\delta_D : D_1\eta_1\delta_D \in SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1))\} .$$

An observation is made on Δ_D as follows: Let Γ_D be the index set of $IM(Ran(y_{i_1}^1)) \times \cdots \times IM(Ran(y_{i_d}^1))$. By definition of

$SBSM(D_1\eta_1(y_{i_1}^{\ 1}, \cdots, y_{i_d}^{\ 1}))$, it follows that for each $l \in \Gamma_D$, if $<S_{i_1}^l, \cdots, S_{i_d}^l> \in IM(Ran(y_{i_1}^{\ 1})) \times \cdots \times IM(Ran(y_{i_d}^{\ 1}))$, then there is the corresponding substitution $\delta_D^l \in \Delta_D$ such that $D_1\eta_1\delta_D^l \in SBSM(D_1\eta_1(y_{i_1}^{\ 1}, \cdots, y_{i_d}^{\ 1}))$ and for each k, $i_1 \leq k \leq i_d$, $Ran(D_1\eta_1\delta_D^l \mid y_k^1) = S_k^l$.

(Case I) Let there be no $d(i+1,0)$-variable in E_1. Then σ_W does not contain any wr-subpair. The pair of clauses which can be resolved among $SBSM(C_1\xi_1(z_{i_1}^{\ 1}, \cdots, z_{i_c}^{\ 1})) \times SBSM(D_1\eta_1(y_{i_1}^{\ 1}, \cdots, y_{i_d}^{\ 1}))$ is first identified. It is done by constructing a set of substitution pairs, denoted by $RESOL^1$, which is a subset of $\Delta_C \times \Delta_D$. $RESOL^1$ is constructed from $\Delta_C \times \Delta_D$ and the wr-mgu σ_W of (2) by the following rules:

(i) A substitution pair $<\delta_C, \delta_D> \in \Delta_C \times \Delta_D$ is a member of $RESOL^1$ if for each wr-substitution component $t/v \in \sigma_W$ it satisfies the condition that

(a) if t is in $C_1\xi_1$ and v is in $D_1\eta_1$, then $Ran(t\,\delta_C) \subseteq Ran(v\,\delta_D)$, or

(b) if t is in $D_1\eta_1$ and v is in $C_1\xi_1$, then $Ran(t\,\delta_D) \subseteq Ran(v\,\delta_C)$.

(ii) No substitution pairs other than those identified by (i) are in $RESOL^1$.

It follows that $RESOL^1$ is not empty.

[The nonemptiness of $RESOL^1$ can be shown as follows: Without loss of generality, let t be in $C_1\xi_1$ and v be in $D_1\eta_1$. Suppose $*$ is used to indicate that k^* means k is either a $d(i,0)$-variable itself or a term containing $d(i,0)$-variable(s) in it. There are four kinds of substitution components in σ_W : t/v, t^*/v, t/v^*, and t^*/v^*. In order to prove the nonemptiness of $RESOL^1$, it suffices to show that for each kind of substitution components the following holds:

(a) $t\,/v$: Since $Ran(t\,\delta_C) = Ran(t)$ and $Ran(v\,\delta_D) = Ran(v)$, *for any* $<\delta_C, \delta_D> \in \Delta_C \times \Delta_D$, $Ran(t\,\delta_C) \subseteq Ran(v\,\delta_D)$.

(b) t^*/v : If t^* is a nonvariable term, then *for any* $<\delta_C, \delta_D> \in \Delta_C \times \Delta_D$, $Ran(t^*\delta_C) \subseteq Ran(v\,\delta_D)$ since $Ran(t^*\delta_C) = Ran(t^*)$ and $Ran(v\,\delta_D) = Ran(v)$. If t^* itself is a $d(i,0)$-variable, then *for any* $S \in IM(Ran(t^*))$, $S \subseteq Ran(v)$ since $S \subseteq Ran(t^*)$ and $Ran(t^*) \subseteq Ran(v)$.

(c) $t\,/v^*$: Since t can not be identical with v^*, $Ran(t) \subset Ran(v^*)$. For any $\delta_C \in \Delta_C$, $Ran(t\,\delta_C) = Ran(t)$. Since v^* is a $d(i,0)$-variable, there is a $S \in IM(Ran(v^*))$ such that $Ran(t) \subseteq S \subset Ran(v^*)$.

(d) t^*/v^* : If t^* is a nonvariable term, then it is an identical case with (c) since $Ran(t^*\delta_C) = Ran(t^*)$ for any $\delta_C \in \Delta_C$. If t^* itself is a $d(i,0)$-variable, there are two possible cases: (i) if $t^* = v^*$, then for any $S_C \in IM(Ran(t^*))$ there is a $S_D \in IM(Ran(v^*))$ such that $S_C = S_D$, or; (ii) if $t^* \neq v^*$ for any $S_C \in IM(Ran(t^*))$ and for any $S_D \in IM(Ran(v^*))$, $S_C \subset S_D$.]

From the way $RESOL^1$ is constructed, it follows that for each $<\delta_C, \delta_D> \in RESOL^1$ and for the two clauses $L_1\xi_1$, $M_1\eta_1$ of (2) $N(L_1\xi_1\delta_C \cup M_1\eta_1\delta_D)$ is unifiable, which further implies that $C_1\xi_1\delta_C \in SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$ and $D_1\eta_1\delta_D \in SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1))$ are resolvable.

Let $<\delta_C, \delta_D> \in RESOL^1$ (notice that there is at least one pair of substitutions in $RESOL^1$ since $RESOL^1$ is not empty). $C_1\xi_1$, $L_1\xi_1$, $D_1\eta_1$ and $M_1\eta_1$ of (2) are considered. It is shown how the two clauses $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$ which are derived by using the $<\delta_C, \delta_D> \in RESOL^1$ are resolved. First, a substitution θ that unifies $N(L_1\xi_1\delta_C \cup M_1\eta_1\delta_D)$ is constructed from σ_W of (2) and the substitution pair $<\delta_C , \delta_D>$ in the following way: For each substitution component $t/v \in \sigma_W$, (i) if t is in $C_1\xi_1$ and v is in $D_1\eta_1$, then $t\delta_C/v\delta_D \in \theta$, or (ii) if t is in $D_1\eta_1$ and v is in $C_1\xi_1$, then $t\delta_D/v\delta_C \in \theta$, and (iii) no other substitution components other than those identified by (i) or (ii) are the elements of θ. Without loss of generality, let t be in $C_1\xi_1$ and let v be in $D_1\eta_1$. Then accordingly there are $t\delta_C$ in $C_1\xi_1\delta_C$ and $v\delta_D$ in $D_1\eta_1\delta_D$ which correspond to t in $C_1\xi_1$ and v in $D_1\eta_1$, respectively. If t and v are unified by σ_W, i.e., $t\sigma_W = v\sigma_W$, then it follows that $t\delta_C$ and $v\delta_D$ are unified by θ since $t\delta_C\theta = t\delta_C$ and $v\delta_D\theta = v\delta_D\{t\delta_C/v\delta_D\} = t\delta_C$. Therefore, it follows that when σ_W unifies $N(L_1\xi_1 \cup M_1\eta_1)$, θ unifies $N(L_1\xi_1\delta_C \cup M_1\eta_1\delta_D)$. Furthermore since the construction of θ from σ_W does not introduce any additional wr-subpair into θ, when σ_W does not contain any wr-subpair, θ does not contain any wr-subpair either. Therefore θ is a Σ-mgu as well as a wr-mgu. To indicate that θ is now a Σ-mgu, let the notation σ_Σ be used for θ. Let E_2 be the Σ-resolvent of $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$ that is generated by using σ_Σ as the Σ-mgu, i.e.,

$$E_2 = (C_1 - L_1)\xi_1\delta_C\sigma_\Sigma \cup (D_1 - M_1)\eta_1\delta_D\sigma_\Sigma .$$

Then from that $C_1\xi_1\delta_C \in SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1)) \subseteq R_\Sigma^i(T_\Sigma)$ and $D_1\eta_1\delta_D \in SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1)) \subseteq R_\Sigma^i(T_\Sigma)$, it follows that $E_2 \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$.

Now $E_1 <> E_2$ is shown. The variables in E_1 of (2) are considered. Since there is no $d(i+1,0)$-variable in E_1 there is no $d(i,0)$-variable in E_1. Therefore, no $d(i,0)$-variable is in $(C_1 - L_1)\xi_1\sigma_W$. This means that although C_1 may contain $d(i,0)$-variables they are eliminated in $(C_1 - L_1)\xi_1\sigma_W$. Consequently, for any variable, say v, in $(C_1 - L_1)\xi_1\sigma_W$, it should hold that either

$v \in \{x_1^1, \cdots, x_n^1\} - \{x_{i_1}^1, \cdots, x_{i_c}^1\}$ or $v \in \{y_1^1, \cdots, y_m^1\} - \{y_{i_1}^1, \cdots, y_{i_d}^1\}$.

Each variable v in $(C_1 - L_1)\xi_1\sigma_W$ is compared with its corresponding variable in $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma$, i.e., $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma | v$, in the following two cases:

(i) When $v \in \{x_1^1, \cdots, x_n^1\} - \{x_{i_1}^1, \cdots, x_{i_c}^1\}$, $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma | v = v\delta_C$.

In this case, $v\delta_C$ is a variable in $C_1\xi_1\delta_C$. By definition of $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1))$, it holds that $Ran(v\delta_C) = Ran(v)$ [by definition, for a clause $C_s \in SBSM(C_1(v_1, \cdots, v_k))$, if u is a variable in C_s and $u \notin \{v_1, \cdots v_k\}$, then $Ran(u) = Ran(C_s | u)$].

(ii) When $v \in \{y_1^1, \cdots, y_m^1\} - \{y_{i_1}^1, \cdots, y_{i_d}^1\}$, $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma | v = v\delta_C\sigma_\Sigma$.

In this case, $v\delta_C\sigma_\Sigma$ is a variable in $D_1\eta_1\delta_D$. By definition of $SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1))$, it holds that $Ran(v\delta_C\sigma_\Sigma) = Ran(v)$.

Therefore, it follows that $(C_1 - L_1)\xi_1\sigma_W <> (C_1 - L_1)\xi_1\delta_C\sigma_\Sigma$. Similarly, it can also be shown that $(D_1 - M_1)\xi_1\sigma_W <> (D_1 - M_1)\xi_1\delta_D\sigma_\Sigma$. It is concluded that $E_1 <> E_2$.

(Case II) Let there be $d(i+1,0)$-variables $e_1, \cdots, e_k$ in E_1. Among these variables some are $d(i+1,i)$-variables and the rest are $d(i,0)$-variables. It is first identified the pairs of clauses which can be resolved among $SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_c}^1)) \times SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_d}^1))$. It is done by constructing a set of substitution pairs, denoted by $RESOL^2$, which is a subset of $\Delta_C \times \Delta_D$. $RESOL^2$ is constructed from $\Delta_C \times \Delta_D$ and the wr-mgu σ_W of (2) by the following rules:

(i) A substitution pair $<\delta_C, \delta_D> \in \Delta_C \times \Delta_D$ is a member of $RESOL^2$ if for each wr-substitution component $t/v \in \sigma_W$ it satisfies the condition that

(a) if $t/v \in \sigma_W$ is not a substitution component constituting a wr-subpair in σ_W, then either $Ran(t\,\delta_C) \subseteq Ran(v\,\delta_D)$ if t is in $C_1\xi_1$ and v is in $D_1\eta_1$ or $Ran(t\,\delta_D) \subseteq Ran(v\,\delta_C)$ if t is in $D_1\eta_1$ and v is in $C_1\xi_1$, or

(b) if $\{t/v_1, t/v_2\}$ is a wr-subpair in σ_W, then either $Ran(v_1\delta_C) \subseteq Ran(v_2\delta_D)$ if v_1 is in $C_1\xi_1$ and v_2 is in $D_1\eta_1$ or $Ran(v_1\delta_D) \subseteq Ran(v_2\delta_C)$ if v_1 is in $D_1\xi_1$ and v_2 is in $C_1\xi_1$.

(ii) No other substitution components other than those identified by (i) are in $RESOL^2$.

It follows that $RESOL^2$ is not empty.

[The nonemptiness of $RESOL^2$ can be shown in a similar way as the nonemptiness of $RESOL^1$ was shown. This time, however, in addition to the four kinds of substitution components shown previously in the proof of nonemptiness of $RESOL^2$, cases for the following four kinds of wr-subpairs are also needed to be considered: $\{t\,/v_1, t\,/v_2\}$, $\{t\,/v_1^*, t\,/v_2\}$, $\{t\,/v_1, t\,/v_2^*\}$, and $\{t\,/v_1^*, t\,/v_2^*\}$.]

From the way $RESOL^2$ is constructed, it follows that for each $<\delta_C, \delta_D> \in RESOL^2$ and for the two clauses $L_1\xi_1$ and $M_1\eta_1$ of (2) $N(L_1\xi_1\delta_C \cup M_1\eta_1\delta_D)$ is unifiable, which further implies that $C_1\xi_1\delta_C \in SBSM(C_1\xi_1(x_{i_1}^{\ 1}, \cdots, x_{i_c}^{\ 1}))$ and $D_1\eta_1\delta_D \in SBSM(D_1\eta_1(y_{i_1}^{\ 1}, \cdots, y_{i_d}^{\ 1}))$ are resolvable.

Now let Γ_e be the index set of $IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$. For each $<S_1^l, \cdots, S_k^l> \in IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$, $l \in \Gamma_e$, let λ^l be a substitution of k substitution components $\{v_1^l/e_1, \cdots, v_k^l/e_k\}$ such that $Ran(v_j^l) = S_j^l$, $1 \le j \le k$. Then $\sigma_W\lambda^l$ also unifies $N(L_1\xi_1 \cup M_1\eta_1)$. Here $E_1\lambda^l$ is the resolvent of $C_1\xi_1$ and $D_1\eta_1$ that is generated by using $\sigma_W\lambda^l$ as the mgu, i.e.,

$$E_1\lambda^l = ((C_1 - L_1)\xi_1\sigma_W \cup (D_1 - M_1)\eta_1\sigma_W)\lambda^l \quad \cdots \text{(3)} .$$

In the rest of the proof, it is shown that for each λ^l , $l \in \Gamma_e$, a clause, say E_2^l , can be derived such that $E_2^l \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$ and $E_1\lambda^l <> E_2^l$.

First, it is shown how E_2^l is derived. Let $e_{i_1}, \cdots, e_{i_h}$ be the $d(i,0)$-variables among $e_1, \cdots, e_k$. Let $v_{i_1}^l, \cdots, v_{i_h}^l$, $\{v_{i_1}^l, \cdots, v_{i_h}^l\} \subseteq \{v_1, \cdots, v_k\}$, be the variables such that $v_{i_j}^l / e_{i_j} \in \lambda^l$, $1 \le j \le h$. Then each element of $\{e_1, \cdots, e_k\} - \{e_{i_1}, \cdots, e_{i_h}\}$ is a $d(i+1,i)$-variable. Let $RESOL^3$ be a subset of $RESOL^2$ such that if $<\delta_C , \delta_D> \in RESOL^3$, then $<\delta_C , \delta_D>$ satisfies the following condition: (i) if e_{i_j} , $1 \le j \le h$, is in $C_1\xi_1$, then $Ran(C_1\xi_1\delta_C \mid e_{i_j}) = Ran(v_{i_j}^l)$, or (ii) if e_{i_j} , $1 \le j \le h$, is in $D_1\eta_1$, then $Ran(D_1\eta_1\delta_D \mid e_{i_j}) = Ran(v_{i_j}^l)$. The preceding conditions are used later in showing $E_1\lambda^l <> E_2^l$.

Let $<\delta_C , \delta_D> \in RESOL^3$. Now a substitution θ is constructed from σ_W of (2), λ^l of (3) and the pair of substitutions $<\delta_C , \delta_D>$ in the following way:

(i) If $t/v \in \sigma_W$ is not a substitution component constituting a wr-subpair in σ_W ,

(a) if t is in $C_1\xi_1$ and v is in $D_1\eta_1$, then $t\delta_C / v\delta_D \in \theta$,

(b) if t is in $D_1\eta_1$ and v is in $C_1\xi_1$, then $t\delta_D / v\delta_C \in \theta$.

(ii) If $\{t/v_1 , t/v_2\}$ is a wr-subpair in σ_W and both or either of v_1 and v_2 is a $d(i,0)$-variable,

(a) if v_1 is in $C_1\xi_1$ and v_2 is in $D_1\eta_1$, then either $v_1\delta_C / v_2\delta_D \in \theta$ if $Ran(v_1\delta_C) \subseteq Ran(v_2\delta_D)$ or $v_2\delta_D / v_1\delta_C \in \theta$ if $Ran(v_2\delta_D) \subseteq Ran(v_1\delta_C)$,

(b) if v_1 is in $D_1\eta_1$ and v_2 is in $C_1\xi_1$, then either $v_1\delta_D/v_2\delta_C \in \theta$ if $Ran(v_1\delta_D) \subseteq Ran(v_2\delta_C)$ or $v_2\delta_C/v_1\delta_D \in \theta$ if $Ran(v_2\delta_C) \subseteq Ran(v_1\delta_D)$.

(iii) If $\{t/v_1, t/v_2\}$ is a wr-subpair in σ_W and neither of v_1 and v_2 is a $d(i,0)$-variable [notice that t is a $d(i+1,i)$-variable], then $\{t'/v_1, t'/v_2\} \subseteq \theta$ where t' is a variable such that if $v/t \in \lambda^l$ where v is some variable then $Ran(t') = Ran(v)$.

(iv) No other substitution component other than those identified by (i), (ii) or (iii) are elements of θ.

In a similar way as was shown in Case I of the inductive step, it can also be shown that θ unifies $N(L_1\xi_1\delta_C \cup M_1\eta_1\delta_D)$ and θ is a Σ-mgu. Let the notation σ_Σ be used for θ to indicate that θ is now a Σ-mgu. Let E_2^l be the resolvent of $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$ that is generated by using σ_Σ as the Σ-mgu, i.e.,

$$E_2^l = (C_1 - L_1)\xi_1\delta_C\sigma_\Sigma \cup (D_1 - M_1)\eta_1\delta_D\sigma_\Sigma .$$

Then from that $C_1\xi_1\delta_C \in SBSM(C_1\xi_1(x_{i_1}^1, \cdots, x_{i_n}^1)) \subseteq R_\Sigma^i(T_\Sigma)$. and $D_1\eta_1\delta_D \in SBSM(D_1\eta_1(y_{i_1}^1, \cdots, y_{i_m}^1)) \subseteq R_\Sigma^i(T_\Sigma)$, it follows that $E_2^l \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$.

Now it is shown that $E_1\lambda^l <> E_2^l$ holds. $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma$ and $(C_1 - L_1)\xi_1\sigma_W\lambda^l$ are considered. It is noticed that $(C_1 - L_1)\xi_1\sigma_W$ subsumes $(C_1 - L_1)\xi_1\sigma_W\lambda^l$ since for each $v/e \in \lambda^l$, $Ran(v) \subseteq Ran(e)$. There are only three kinds variables in $(C_1 - L_1)\xi_1\sigma_W$: $d(i,0)$-variables, $d(i+1,i)$-variables, and non-$d(i+1,0)$-variables. For each kind of these variables the following holds:

(i) For any $d(i,0)$-variable $e_{i_s} \in \{e_{i_1}, \cdots, e_{i_k}\}$, if e_{i_s} is in $(C_1 - L_1)\xi_1\sigma_W$, then, from the way that $RESOL^3$ is constructed, it follows that $Ran((C_1 - L_1)\xi_1\delta_C\sigma_\Sigma \mid e_{i_s}) = Ran(v^l_{i_s}) = Ran((C_1 - L_1)\xi_1\sigma_W\lambda^l \mid e_{i_s})$ where $v^l_{i_s} \in \{v^l_{i_1}, \cdots, v^l_{i_k}\}$ is the variable such that $v^l_{i_s}/e_{i_s} \in \lambda^l$.

(ii) For each $d(i+1,i)$-variable $e_{i_t} \in \{e_1, \cdots, e_k\} - \{e_{i_1}, \cdots, e_{i_k}\}$, if e_{i_t} is in $(C_1 - L_1)\xi_1\sigma_W$, then from the way that σ_Σ (i.e., θ) is constructed [see (iii) in the constructing stage of θ], it follows that $Ran((C_1 - L_1)\xi_1\delta_C\sigma_\Sigma \mid e_{i_t}) = Ran(v^l_{i_t}) = Ran((C_1 - L_1)\xi_1\sigma_W\lambda^l \mid e_{i_t})$ where $v^l_{i_t} \in \{v^l_{i_1}, \cdots, v^l_{i_k}\}$ is the variable such that $v^l_{i_s}/e_{i_s} \in \lambda^l$.

(iii) For each non-$d(i+1,0)$-variable, say u, if u is in $(C_1 - L_1)\xi_1\sigma_W$, then $Ran((C_1 - L_1)\xi_1\delta_C\sigma_\Sigma \mid u) = Ran(u) = Ran((C_1 - L_1)\xi_1\sigma_W\lambda^l \mid u)$ since $C_1\xi_1\delta_C \in SBSM(C_1(x_1^1, \cdots, x_n^1))$ and $u\lambda^l = u$.

Therefore $(C_1 - L_1)\xi_1\delta_C\sigma_\Sigma <> (C_1 - L_1)\xi_1\sigma_W\lambda^l$. Similar arguments can be applied to $(D_1 - M_1)\xi_1\delta_D\sigma_\Sigma$ and $(D_1 - M_1)\xi_1\sigma_W\lambda^l$ to conclude that $(D_1 - M_1)\xi_1\delta_D\sigma_\Sigma <> (D_1 - M_1)\xi_1\sigma_W\lambda^l$. Consequently, it follows that $E_1\lambda^l <> E_2^l$. This conclusion is sufficient enough to say that $E_1\lambda^l$ subsumes E_2^l because for each $v/e \in \lambda^l$, $Ran(v) \Subset Ran(e)$. Since the preceding argument has been made for each λ^l, $l \in \Gamma_e$, the following is finally concluded: for each $<S_1^l, \cdots, S_k^l> \in IM(Ran(e_1)) \times \cdots \times IM(Ran(e_k))$, $l \in \Gamma_e$, there is a clause $E_2^{l(j)} \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$ such that (i) E_1 subsumes $E_2^{l(j)}$, (ii) $Ran(E_2^l \mid e_h) = S_j^h$, $1 \le h \le k$, and (iii) for any variable v in E_1 other than $e_1, \cdots, e_k$, $Ran(E_2^l \mid v) = Ran(v)$. Hence it follows that there is a $SBSM(E_1(e_1, \cdots, e_k)) \subseteq R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$. *Q.E.D.*

Example 12.2.2

Consider Example 12.1.1. The case when there is no $d(i+1,0)$-variable in E_1 is considered. Let E_1 be $R(f^F(z^{\Sigma E}),f^H(z^{\Sigma E})) \in R_W^2(T_\Sigma) - R_W^1(T_\Sigma)$. No $d(2,0)$-variable is in E_1. Let E_1 be a wr-resolvent of $\neg Q(x^{\Sigma E},f^H(x^{\Sigma E})) \in R_W^1(T_\Sigma)$ and $Q(x^{\Sigma K},y^{\Sigma E}) \cup R(f^F(x^{\Sigma K}),y^{\Sigma E}) \in R_W^1(T_\Sigma)$. Let the two resolvents be C_1 and D_1, respectively. The variable $x^{\Sigma K}$ in D_1 is a $d(1,0)$-variable. There is a $SBSM(D_1(x^{\Sigma K})) \subseteq R_\Sigma^1(T_\Sigma)$ such as

$$SBSM(D_1(x^{\Sigma K})) = \{Q(x^{\Sigma D},y^{\Sigma E}) \cup R(f^F(x^{\Sigma D}),y^{\Sigma E}),\ Q(x^{\Sigma E},y^{\Sigma E}) \cup R(f^F(x^{\Sigma E}),y^{\Sigma E})\}.$$

Two clauses $\neg Q(x^{\Sigma D},f^F(x^{\Sigma E}))$, $Q(x^{\Sigma E},y^{\Sigma E}) \cup R(f^F(x^{\Sigma E}),y^{\Sigma E}) \in R_\Sigma^1(T_\Sigma)$ are considered. Let these two clauses be C_2 and D_2, respectively. It is noticed that $D_2 \in SBSM(D_1(x^{\Sigma K}))$. A Σ-resolvent, say E_2, of C_2 and D_2 is

$$E_2 = R(f^F(w^{\Sigma F}),f^H(w^{\Sigma F})).$$

It is clear that $E_2 \in R_\Sigma^2(T_\Sigma) - R_\Sigma^1(T_\Sigma)$ and $E_2 <> E_1$.

Now the case when there are some $d(i+1,0)$-variables in E_1 is considered. Let E_1 be $Q(x^{\Sigma K},x^{\Sigma I}) \in R_W^2(T_\Sigma) - R_W^1(T_\Sigma)$. The two variables $x^{\Sigma K}$, $x^{\Sigma I}$ in E_1 are $d(2,0)$-variables. Let E_1 be a wr-resolvent of two clauses $\neg P(x^{\Sigma C})$, $P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma I}) \in R_W^1(T_\Sigma)$. Let these two clauses be C_1 and D_1, respectively. The variable $x^{\Sigma I}$ in D_1 is a $d(1,0)$-variable. It is seen that there is a $SBSM(D_1(x^{\Sigma I})) \subseteq R_\Sigma^1(T_\Sigma)$ such as

$$SBSM(D_1(x^{\Sigma I})) = \{P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma F}),\ P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma G}),\ P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma H})\}.$$

Let C_2 be $\neg P(x^{\Sigma C}) \in R_{\Sigma}^{1}(T_{\Sigma})$. Let D_2^1, D_2^2, and D_2^3 be $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma F})$, $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma G})$, $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma H}) \in R_{\Sigma}^{1}(T_{\Sigma})$, respectively. It is noticed that $D_2^i \in SBSM(D_1(x^{\Sigma I}))$, $1 \leq i \leq 3$. A Σ-resolvent is derived from C_2 and each D_2^i, $1 \leq i \leq 3$, as follows: if the resolution operator $R_{\Sigma}(\cdot)$ is used,

$$\begin{aligned} R_{\Sigma}(C_2, D_2^1) &= \{Q(w^{\Sigma D}, w^{\Sigma F}), Q(w^{\Sigma E}, w^{\Sigma F})\}, \\ R_{\Sigma}(C_2, D_2^2) &= \{Q(w^{\Sigma D}, w^{\Sigma G}), Q(w^{\Sigma E}, w^{\Sigma G})\}, \\ R_{\Sigma}(C_2, D_2^3) &= \{Q(w^{\Sigma D}, w^{\Sigma H}), Q(w^{\Sigma E}, w^{\Sigma H})\}. \end{aligned}$$

Let $E_2 = \bigcup_{1 \leq i \leq 3} R_{\Sigma}(C_2, D_2^i)$. It is clear that E_2 is a $SBSM(E_1(x^{\Sigma K}, x^{\Sigma I}))$ and $E_2 \subseteq R_{\Sigma}^{2}(T_{\Sigma}) - R_{\Sigma}^{1}(T_{\Sigma})$.

A corollary follows to Lemma 12.2.3 which assures that the length of the shortest deduction sequence for $R_{\Sigma}(\cdot)$ is not longer than that of $R_W(\cdot)$.

<u>Corollary 12.2.4</u>

Given a $< OA, T_{\Sigma} >$, if n is the smallest non-negative integer for which $R_W^n(T_{\Sigma})$ contains $\Box$, then $R_{\Sigma}^n(T_{\Sigma})$ also contains $\Box$.

Proof. Let n be the smallest non-negative integer for which $R_W^n(T_{\Sigma})$ contains $\Box$. Then $\Box$ is a resolvent of two clauses in $R_W^{n-1}(T_{\Sigma})$, say C_1, D_1. There are three possible cases: (i) there is no $d(n-1,0)$-variable in either C_1 or D_1, (ii) some $d(n-1,0)$-variables are in either C_1 or D_1, and (iii) some $d(n-1,0)$-variables are in both C_1 and D_1. Again only the most general case is considered: case (iii). Since

$\square$ is a resolvent of C_1 and D_1, the following holds: C_1 and D_1 are singletons and there is a wr-mgu, say σ_W , unifying $N(C_1\xi_1 \cup D_1\eta_1)$. Then it follows that

$$C_1\xi_1\sigma_W \cup D_1\eta_1\sigma_W = \square,$$

where $C_1\xi_1\sigma_W$ and $D_1\eta_1\sigma_W$ are singletons whose respective members are complements.

Since $\square$ does not have any $d(n,0)$-variables in it, the proof here is similar to the proof of Lemma 12.2.3 which was shown for the case when there is no $d(n,0)$-variable in E_1. Let $x_1^1, \cdots, x_l^1$ and $y_1^1, \cdots, y_m^1$ be the $d(n-1,0)$-variables in C_1 and D_1, respectively. By Lemma 12.2.3, there are $SBSM(C_1(x_1^1, \cdots, x_l^1))$, $SBSM(D_1(y_1^1, \cdots, y_m^1)) \subseteq R_\Sigma^{n-1}(T_\Sigma)$. Let Δ_C and Δ_D be defined in the same way as Δ_C and Δ_D was defined in the proof of Lemma 12.2.3. Let $<\delta_C, \delta_D> \in \Delta_C \times \Delta_D$. $C_1\xi_1$ and $D_1\eta_1$ subsume $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$, respectively. As was shown in the proof of Lemma 12.2.3, it can be shown that there is a Σ-mgu, say σ_Σ, unifying $N(C_1\xi_1\delta_C \cup D_1\eta_1\delta_D)$. From the fact that $C_1\xi_1$ and $D_1\eta_1$ subsume $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$, respectively, it follows that $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$ must be singletons. From the facts that $C_1\xi_1\delta_C$ and $D_1\eta_1\delta_D$ are singletons and that σ_Σ unifies $N(C_1\xi_1\delta_C \cup D_1\eta_1\delta_D)$, it immediately follows that

$$C_1\xi_1\delta_C\sigma_\Sigma \cup D_1\eta_1\delta_D\sigma_\Sigma = \square.$$

$R_\Sigma^n(T_\Sigma) - R_\Sigma^{n-1}(T_\Sigma)$ contains $\square$. So does $R_\Sigma^n(T_\Sigma)$. *Q.E.D.*

From Corollary 12.2.2 and Corollary 12.2.4, the following is concluded:

Theorem 12.2.5

Given $< OA \ , T_\Sigma >$, if n is the smallest non-negative integer for which $R_W^n(T_\Sigma)$ contains $\square$ and m is the smallest non-negative integer for which $R_\Sigma^m(T_\Sigma)$ contains $\square$, then $n = m$.

Proof. The theorem immediately follows from Corollary 12.2.2 and Corollary 12.2.4. *Q.E.D.*

The preceding result is the conclusion of the firsts stage, i.e., given a many-sorted theory $< OA \ , T_\Sigma >$ the length of the shortest refutation generated by $R_W(\cdot)$ is identical with that generated by $R_\Sigma(\cdot)$. The preceding result is illustrated by an example at the end of this section.

The second stage of the comparison of $R_W(\cdot)$ and $R_\Sigma(\cdot)$ is now given. First it is shown that, given a many-sorted theory $< OA \ , T_\Sigma >$, the number of wr-resolutions generated by $R_W(\cdot)$ at each level is smaller or equal to that generated by $R_\Sigma(\cdot)$ at the same level.

Lemma 12.2.6

Given a $< OA \ , T_\Sigma >$, for each $i \geq 0$,

$$| R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma) | \leq | R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma) | \ .$$

Proof. This result is an immediate consequence of Lemma 12.2.3. For each $i \geq 0$, let E_1 be a clause in $R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$. Let there be no $d(i+1,0)$-

variable in E_1. By Lemma 12.2.3, there is a clause $E_2 \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$ such that $E_1 <> E_2$. Let there be some $d(i+1,0)$-variables $e_1, \cdots, e_k$ in E_1. By Lemma 12.2.3, there is a $SBSM(E_1(e_1, \cdots, e_k)) \subseteq R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$. It follows that

$$| SBSM(E_1(e_1, \cdots, e_k)) | \leq | IM(Ran(e_1)) | \times \cdots \times | IM(Ran(e_k)) | .$$

Since $| IM(Ran(e_i)) | > 1$, $1 \leq i \leq k$, $| SBSM(E_1(e_1, \cdots, e_k)) | > 1$. Since for each $E_1 \in R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)$ there is either a corresponding clause $E_2 \in R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$ such that $E_1 <> E_2$ or a corresponding set of clauses $SBSM(E_1(e_1, \cdots, e_k)) \subseteq R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)$, it holds that

$$| R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma) | \leq | R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma) | .$$

Q.E.D.

The overall efficiency issue is concluded by the following theorem:

Theorem 12.2.7

Given $< OA , T_\Sigma >$, if n is the smallest non-negative integer for which $R_W^n(T_\Sigma)$ and $R_\Sigma^n(T_\Sigma)$ both contain $\square$, then $| R_W^n(T_\Sigma) | \leq | R_\Sigma^n(T_\Sigma) |$.

Proof. This result immediately follows from Lemma 12.2.6. *Q.E.D.*

The preceding theorem indicates that $R_W(\cdot)$ is more efficient than $R_\Sigma(\cdot)$. The result of Theorem 12.2.7 is illustrated in the following example.

Example 12.2.3

Consider the following many-sorted theory $< OA \;, T_\Sigma >$:

OA : (0.1) $D \subseteq B \;,\; D \subseteq C \;,$

(0.2) $E \subseteq B \;,\; E \subseteq C \;,$

(0.3) $F \subseteq D \;,\; F \subseteq E \;,$

(0.4) $G \subseteq D \;,\; G \subseteq E \;,$

(0.5) $H \subseteq D \;,\; H \subseteq E \;,$

(0.6) $I \subseteq E \;,$

T_Σ : (0.7) $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma E}) \cup R(f^F(x^{\Sigma B}), x^{\Sigma E}) \cup W(x^{\Sigma B}) \;,$

(0.8) $\neg P(x^{\Sigma C}) \;,$

(0.9) $\neg Q(x^{\Sigma E}, g^H(x^{\Sigma E})) \;,$

(0.10) $\neg R(x^{\Sigma E}, x^{\Sigma D}) \;,$

(0.11) $\neg W(x^{\Sigma I}) \;.$

Here a pair of numbers $(i \;.\; j)$ is used to identify each clause. Each clause preceded by $(i \;.\; j)$ is the j^{th} clause at the i^{th} resolution operation. For example, for $i = 0$, $R_W^0(T_\Sigma) = \{(0.7)\,, \cdots, (0.11)\}$. Same notation is also used to identify the clauses for the Σ-resolution $R_\Sigma(\,\cdot\,)$, i.e., $R_\Sigma^0(T_\Sigma) = \{(0.7)\,, \cdots, (0.11)\}$. It is shown which of $R_W(\,\cdot\,)$ and $R_\Sigma(\,\cdot\,)$ is more efficient by generating $R_W^n(T_\Sigma)$ and $R_\Sigma^n(T_\Sigma)$ each of which contains $\Box$. When the members of $R_W^n(T_\Sigma)$ and $R_\Sigma^n(T_\Sigma)$ are appropriately aligned, they are two refutations, one generated by $R_W(\,\cdot\,)$ and the other generated by $R_\Sigma(\,\cdot\,)$, respectively. Both refutations are so lengthy that their complete sequencies are shown in Appendix C.

The results obtained from the two refutations are summarized in the following table that shows the numbers of resolvents generated at each level:

Comparison of $R_W(\cdot)$ and $R_\Sigma(\cdot)$

Level	No. of Resolvents Generated	
	$R_W(\cdot)$	$R_\Sigma(\cdot)$
0	5	5
1	4	7
2	12	29
3	12	24
4	4	5
Total	37	70

The following observations are made from the table: first, in both refutations, $\square$ turns up at the same level, i.e., at level 4 (cf. Theorem 12.2.5); second, at each level i, $0 \leq i \leq 4$, $|R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma)| \leq |R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma)|$ (cf. Theorem 12.2.6), and; third $|R_W^4(T_\Sigma)| \leq |R_\Sigma^4(T_\Sigma)|$ (cf. Theorem 12.2.7). More details about the refutations can be found in Appendix C; for example, what unary predicate symbols are dynamically defined in the refutation by $R_W(\cdot)$ and which Σ-resolvents are indeed useless.

12.3. Conclusions and Future Work

First, a problem was identified that might occur in the currently known many-sorted resolution, such as the one reported by Walther. This problem can be avoided if new sorts are introduced while the resolution is being carried out. However, doing so is not possible if the theory to be refuted is expressed in an ordinary many-sorted language and the language is not to be revised along the way refutation is carried out.

To alleviate such a situation, the language called one-sorted language with aggregate variables (L_{Σ}^{1}), which is obtained by embedding aggregate variables in a one-sorted language, was proposed. A many-sorted theory was then expressed in L_{Σ}^{1} and a new approach, called UWR-resolution, was presented. In this resolution new sorts are dynamically introduced as needed using the aggregate variables. The completeness of this resolution was shown and the efficiency of the resolution was discussed.

The preceding approach that has been shown throughout Part II is not the only way of embodying the idea of unifying a pair of variables satisfying a certain condition over the weakest range. Alternative approaches are available. Two alternative approaches are discussed in Appendix D.

There are two ways of extending the work discussed so far. One way is to study what extension should be made if the theory to be refuted by the UWR-resolution is expressed in the language $L_{\Sigma}^{=}$ which is obtained by including the "equality symbol" in L_{Σ}^{1}. In this case, it is expected that an inference rule, what is often called "paramodulation", must be additionally introduced. The other way is to study the effect of combining with the UWR-resolution various control strategies used in a one-sorted resolution. These strategies include those used in the one-sorted resolution such as lock resolution, semantic resolution, linear resolution, and unit resolution.

CHAPTER XIII

CONCLUSIONS

The implications resulting from the two applications, one in Part I and the other in Part II, are two fold: extending the first-order predicate calculi by embedding aggregate variables in their languages is theoretically sound and the extended calculi are practically useful. These two implications are summarized in this chapter.

First, the theoretic soundness of the extended calculi is summarized. The two languages for the first-order predicate calculi, a one-sorted language and a many-sorted language, were extended by embedding a new type of syntactic object, called *aggregate variables*. The aggregate variables are syntactically ordinary sort variables, but semantically they are variables whose ranges are restricted to unary relations instead of sorts. Therefore, whenever aggregate variables are introduced, the sort structure determined a priori remains intact, although the system itself is augmented by new unary relations that will be the aggregate variables range of interpretation, which is the process known as expansion by definitions (e.g., [Shoe67]). This property of the extended predicate calculus is called *Σ-extensibility.* The Σ-extensibility of L_Σ and the Σ-extensibility of L_Σ^1 were shown in the form of theorems. These Σ-extensibilities of the extended calculi assure that one of the problems of an ordinary many-sorted language, namely, the inflexible usage of sort variables (e.g., [Cohn83]), can now be overcome.

In the rest, the practical usability of the extended calculi is discussed. The Σ-extensibilities of the extended calculi implied the flexible usage of aggregate variables (contrasted to the inflexibility of the ordinary sort variables) which led to two applications, one in the distributed database design area and the other in the automatic theorem proving area. Based on these two applications, the practical usability of the extended calculi can be generalized.

Before such generalization is made, the significance of the extended calculi in each application is reviewed. The significance of L_Σ in the KBDDBS design is given first. The significance of L_Σ in the KBDDBS is twofold: (i) L_Σ provides more compact expressive power than does L_m and, therefore, (ii) to a certain extent, it became possible to develop a simple syntactic matching process as the inference procedure involving specific formulas in L_Σ. The compact expressive power of L_Σ over L_m was due to the fact that L_Σ permitted the introduction of the aggregate variable whose ranges were restricted to subsets of sort domains, something that could not be done in L_m . This compact expressive power of L_Σ allowed an easy way of endowing dual semantics† to the formulas of L_Σ a la [Kowa74], which otherwise might not have been possible. The knowledge about the data was able to be expressed in a special form called the Σ-Horn formula, queries were expressed in the Σ-normal form, and a syntactic matching process was able to be developed as the inference procedure with which the knowledge of the Σ-Horn form was applied to the user queries in Σ-normal form in an inferencing manner.

† In [Mylo81] Mylopoulos mentions "An interesting departure from logical representation schemes has been proposed by Kowalski [Kowa74] who argues in favor of a *dual semantics* for logical formulas of the form $B_1 \cap B_2 \cap \cdots \cap B_n \rightarrow A$. The first is the traditional Tarskian semantics. The second is a procedural semantics which interprets the formula as "If you want to establish A, try to establish B_1 and B_2 and $\cdots$ and B_n" ".

Now the significance of L_{Σ}^{1} in the UWR-resolution is discussed. The significance of L_{Σ}^{1} in the UWR-resolution is that L_{Σ}^{1} allows the introduction of the variables ranging over the intersection of two specific sorts determined previously in the middle of refutation. Introducing variables over such a sort could also have been done even if the theory to be refuted is expressed in L_m , since all the likely-to-be-used sorts can be determined before refutation begins, so the sort structure of the theory can be modified to include the all the likely-to-be-used sorts. The problem here was that unnecessary additional axioms for the theory must be generated for no usage. This problem will not be encountered if a variable ranging over a new sort is dynamically introduced using aggregate variables.

Based on these two applications, the practical usabilities of the extended calculi can be generalized to a certain extent. The following situation is considered: (i) a system involving more than one category of objects is axiomatized, (ii) a need arises to introduce a variable ranging over a sort that does not exist in the sort structure determined by the categories of the objects, and (iii) it is not desired to change the sort structure determined a priori. In this situation, the system can be axiomatized based on the proposed extended calculi; variables ranging over new sorts can be introduced as needed.

APPENDICES

APPENDIX A

A Relational Database Example

An Auto Corporation Database

DIVISIONS

div#	div_name	head
01AP	Buick	Patrick
01HQ	Finance	Joyol
01PP	Elextra	Shin
02AP	Pontiac	Lee
02PP	Body	Rieger
03PP	Engina	Meltzer
03PP	Trans	Frege
04AP	Frantana	Frege
05AP	Omnus	Gelperin

DEALERS

d#	address	d_type
01A	Ann Arbor	51
03A	Dearborn	30
07A	Flint	50
26M	Cleveland	20
33B	Cleveland	30
48B	Rockford	31
55L	Flint	51
65B	Detroit	20
66L	Niles	23
70A	Lansing	70

ITEMS

item#	i_name	i_type
A01	ink	stationery
A02	note pad	stationery
B47	Eland	bus
C05	blue 5	paint
C06	white 7	paint
N11	square 11"	nut
P02	distributer	engine part
P03	radiator	engine part
S01	micro proc.	elect. part
S02	battery	elect. part
V01	Astre	sedan
V03	Camaro	sedan
W09	Cabriolet	van
X77	iron 7"	plate
X89	iron 9"	plate

SALES

div#	d#	item#
01AP	01A	V01
01AP	07A	W09
01PP	55L	S01
01PP	07A	P02
02AP	01A	B47
02PP	03A	P02
03PP	01A	P03
03PP	03A	S02
04AP	01A	V03
05AP	55L	V03
05PP	55L	S02

APPENDIX B

An Intermediate L^1_Σ-Version of the Herbrand Theorem

An intermediate L^1_Σ-version of the Herbrand theorem is derived. In [KrKr67], the following form of the Herbrand theorem is given as an exercise along with its solution.

> 3. Refinement of the Uniformity Theorem (for predicate calculus with several types of variables).
>
> a) Show that if $\exists x_1 \cdots \exists x_n A$, where A is quantifier free, is a theorem then there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$ $(1 \leq i \leq p)$ of n-tuples of terms of the language of A such that $A_1 \cup \cdots \cup A_p$ is a theorem, where A_i is obtained by replacing x_j in A by $t_j^{(i)}$.

In the preceding statements the following notions are used: Let $L(A)$ be the language of A † with k types (or sorts in this context) of objects. Then there are k infinite disjoint sets $V^1, \cdots, V^k$ where the elements of V^i , $1 \leq i \leq k$, are called variables of type i of $L(A)$. Then each variable x_j , $1 \leq j \leq n$, belongs to some V^i , $1 \leq i \leq k$. Let $Term$ be the set of terms of $L(A)$. $Term$ is divided into k disjoint sets $Term_1, \cdots, Term_k$. Then each term $t_j^{(i)}$ $(1 \leq i \leq p, 1 \leq j \leq n)$ belongs to some $Term_l$, $1 \leq l \leq k$.

This theorem only states the necessary part of the condition. If the sufficient part of the condition is also combined, the preceding exercise can be rephrased in the following form of a theorem:

---

† By the language of a formula, it is meant the language whose variables are those of L and whose relations and function symbols are those which occur in formula A .

Theorem 1†

Let $A(x_1, \cdots, x_n)$ be a quantifier free formula with free variables $x_1, \cdots, x_n$. Then $\exists x_1 \cdots \exists x_n\, A(x_1, \cdots, x_n)$ is a theorem if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$ $(1 \leq i \leq p)$ of n-tuples of terms of the language of A such that $A_1 \cup \cdots \cup A_p$ is a theorem, where A_i is obtained by replacing x_j, $1 \leq j \leq n$, in A by $t_j^{(i)}$.

The formalism used in the theorem proving is based on the notions of unsatisfiability and refutation rather than upon the notions of validity and proof. The following dual form of the theorem must be derived.

Theorem 2

Let $A(x_1, \cdots, x_n)$ be a quantifier free formula with free variables $x_1, \cdots, x_n$. Then $\forall x_1 \cdots \forall x_n\, A(x_1, \cdots, x_n)$ is unsatisfiable if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$ $(1 \leq i \leq p)$ of n-tuples of terms of the language of A such that $A_1 \cap \cdots \cap A_p$ is unsatisfiable where A_i is obtained by replacing x_j, $1 \leq j \leq n$, in A by $t_j^{(i)}$.

Proof. It is sufficient to put $A(x_1, \cdots, x_n) = \neg B(x_1, \cdots, x_n)$ where $B(x_1, \cdots, x_n)$ is a quantifier formula with free variables $x_1, \cdots, x_n$. Then by Theorem 1 and $B(x_1, \cdots, x_n) = \neg A(x_1, \cdots, x_n)$, it holds that

† Kleene adequately points out what the Herbrand theorem indicates. Quoting Kleene [Klee67], "We may summarize Herbrand's theorem by saying that it reduces the question of the provability of a particular formula with quantifiers (in the first instance, a prenex formula) to the question of the validity (or provability) in the propositional calculus of some one of a countably infinite class of quantifier-free formulas (the Herbrand disjunctions)."

$\exists x_1 \cdots \exists x_n \neg A(x_1, \cdots, x_n)$ is a theorem if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$ $(1 \leq i \leq p)$ of n-tuples of terms of the language of A such that $\neg A_1 \cup \cdots \cup \neg A_p$ is a theorem. It immediately follows that $\forall x_1 \cdots \forall x_n A(x_1 \cdots x_n)$ is unsatisfiable if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$ $(1 \leq i \leq p)$ of n-tuples of terms of the language of A such that $A_1 \cap \cdots \cap A_p$ is unsatisfiable. *Q.E.D.*

Now, based on Theorem 9.3.1, the formula A can be expressed in L^1_Σ. That is, the language of A, denoted by $L^1_\Sigma(A)$, is the language whose variables are those of L^1_Σ and whose relations and function symbols are those that occur in formula A. Finally, Theorem 2 can be rephrased in the following form.

<u>Theorem 3</u>

Let $A(x_1, \cdots, x_n)$ be a quantifier free formula with free variables $x_1, \cdots, x_n$. Then $\forall x_1 \cdots \forall x_n A(x_1, \cdots, x_n)$ is unsatisfiable if and only if there is a sequence $(t_1^{(i)}, \cdots, t_n^{(i)})$, $1 \leq i \leq p$, of n-tuples of terms of $L^1_\Sigma(A)$ such that $A_1 \cap \cdots \cap A_p$ is unsatisfiable where A_i is obtained by replacing x_j, $1 \leq j \leq n$, in A by $t_j^{(i)}$.

APPENDIX C

Refutations by $R_W(\cdot)$ and $R_\Sigma(\cdot)$

Two refutations of a given many-sorted theory are shown. One is generated by $R_W(\cdot)$ and the other is generated by $R_\Sigma(\cdot)$. Consider the following many-sorted theory $< OA\ , T_\Sigma >$ given in Example 12.2.3:

OA : (0.1) $D \Subset B\ ,\ D \Subset C\ ,$

(0.2) $E \Subset B\ ,\ E \Subset C\ ,$

(0.3) $F \Subset D\ ,\ F \Subset E\ ,$

(0.4) $G \Subset D\ ,\ G \Subset E\ ,$

(0.5) $H \Subset D\ ,\ H \Subset E\ ,$

(0.6) $I \Subset E\ ,$

T_Σ : (0.7) $P(x^{\Sigma B}) \cup Q(x^{\Sigma B}, x^{\Sigma E}) \cup R(f^F(x^{\Sigma B}), x^{\Sigma E}) \cup W(x^{\Sigma B})\ ,$

(0.8) $\neg\, P(x^{\Sigma C})\ ,$

(0.9) $\neg\, Q(x^{\Sigma E}, g^H(x^{\Sigma E}))\ ,$

(0.10) $\neg\, R(x^{\Sigma E}, x^{\Sigma D})\ ,$

(0.11) $\neg\, W(x^{\Sigma I})\ .$

First the refutation of $< OA\ , T_\Sigma >$ generated by $R_W(\cdot)$ is shown and then the refutation generated by $R_\Sigma(\cdot)$ is shown. Remember both $R_W(\cdot)$ and $R_\Sigma(\cdot)$ are level-saturation schemes. Each refutation is an alignment of the resolvents generated at each level, i.e., a sequence. In each sequence, the first column shows the numbering for the deduction sequence. The numbering stops when the first $\square$ turns up. The second column contains the identifier of each resolvent. The first digit of each identifier indicates the level at which the resolvent is generated. The third column contains the identifiers of the parent clauses of their corresponding resolvent. The fourth column shows the resolvents. The fifth column is used to show the dynamically introduced sorts if the refutation is the one by $R_W(\,\cdot\,)$. If the refutation is the one by $R_\Sigma(\,\cdot\,)$, then it is used to indicate useless resolvents.

[1] **The refutation generated by $R_W(\,\cdot\,)$**

ded. seq.	res. id.	parent clauses	resolvents	note
1	(1.1)	(0.7),(0.8)	$Q(x^{\Sigma K},x^{\Sigma E}) \cup R(f^F(x^{\Sigma K}),x^{\Sigma E}) \cup W(x^{\Sigma K})$	$K \leftrightarrows B \cap C$
2	(1.2)	(0.7),(0.9)	$P(y^{\Sigma E}) \cup R(f^F(y^{\Sigma E}),g^H(y^{\Sigma E})) \cup W(x^{\Sigma E})$	
3	(1.3)	(0.7),(0.10)	$P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma J}) \cup W(x^{\Sigma B})$	$J \leftrightarrows D \cap E$
4	(1.4)	(0.7),(0.11)	$P(x^{\Sigma J}) \cup Q(x^{\Sigma J},x^{\Sigma E}) \cup R(f^F(x^{\Sigma J}),x^{\Sigma E})$	
5	(2.1)	(1.1),(0.9)	$R(f^F(y^{\Sigma E}),x^{\Sigma E}) \cup W(y^{\Sigma E})$	
6	(2.2)	(1.1),(0.10)	$Q(x^{\Sigma K},x^{\Sigma J}) \cup W(x^{\Sigma K})$	K , J
7	(2.3)	(1.1),(0.11)	$Q(x^{\Sigma J},x^{\Sigma E}) \cup R(f^F(x^{\Sigma J}),x^{\Sigma E})$	
8	(2.4)	(1.2),(0.8)	same as (2.1)	
9	(2.5)	(1.2),(0.10)	$P(y^{\Sigma E}) \cup W(x^{\Sigma E})$	
10	(2.6)	(1.2),(0.11)	$P(y^{\Sigma J}) \cup R(f^F(y^{\Sigma J}),g^H(y^{\Sigma J}))$	
11	(2.7)	(1.3),(0.8)	same as (2.2)	
12	(2.8)	(1.3),(0.9)	same as (2.5)	
13	(2.9)	(1.3),(0.11)	$P(x^{\Sigma J}) \cup Q(x^{\Sigma J},x^{\Sigma J})$	J
14	(2.10)	(1.4),(0.8)	same as (2.3)	
15	(2.11)	(1.4),(0.9)	same as (2.6)	
16	(2.12)	(1.4),(0.10)	same as (2.9)	
17	(3.1)	(2.1),(0.10)	$W(y^{\Sigma E})$	
18	(3.2)	(2.1),(0.11)	$R(f^F(y^{\Sigma J}),g^H(y^{\Sigma J}))$	
19	(3.3)	(2.2),(0.9)	same as (3.1)	
20	(3.4)	(2.2),(0.11)	$Q(x^{\Sigma J},x^{\Sigma J})$	J
21	(3.5)	(2.3),(0.9)	same as (3.2)	
22	(3.6)	(2.3),(0.10)	same as (3.4)	
23	(3.7)	(2.5),(0.8)	same as (3.3)	
24	(3.8)	(2.5),(0.11)	$P(y^{\Sigma J})$	
25	(3.9)	(2.6),(0.8)	same as (3.2)	
26	(3.10)	(2.6),(0.10)	same as (3.8)	
27	(3.11)	(2.9),(0.8)	same as (3.4)	
28	(3.12)	(2.9),(0.9)	same as (3.8)	
29	(4.1)	(3.1),(0.11)	□	
	(4.2)	(3.2),(0.10)	□	
	(4.3)	(3.4),(0.9)	□	
	(4.4)	(3.8),(0.8)	□	

[2] **The refutation generated by** $R_\Sigma(\cdot)$

ded. seq.	res. id.	parent clauses	resolvents	note
1	(1.1a)	(0.7),(0.8)	$Q(y^{\Sigma D},x^{\Sigma E}) \cup R(f^F(y^{\Sigma D}),x^{\Sigma E}) \cup W(y^{\Sigma D})$	useless
2	(1.1b)	(0.7),(0.8)	$Q(y^{\Sigma E},x^{\Sigma E}) \cup R(f^F(y^{\Sigma E}),x^{\Sigma E}) \cup W(y^{\Sigma E})$	
3	(1.2)	(0.7),(0.9)	$P(y^{\Sigma E}) \cup R(f^F(y^{\Sigma E}),g^H(y^{\Sigma E})) \cup W(x^{\Sigma E})$	
4	(1.3a)	(0.7),(0.10)	$P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma F}) \cup W(x^{\Sigma B})$	useless
5	(1.3b)	(0.7),(0.10)	$P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma G}) \cup W(x^{\Sigma B})$	useless
6	(1.3c)	(0.7),(0.10)	$P(x^{\Sigma B}) \cup Q(x^{\Sigma B},x^{\Sigma H}) \cup W(x^{\Sigma B})$	
7	(1.4)	(0.7),(0.11)	$P(x^{\Sigma I}) \cup Q(x^{\Sigma I},x^{\Sigma E}) \cup R(f^F(x^{\Sigma I}),x^{\Sigma E})$	
8	(2.1a)	(1.1a),(0.9)	$R(f^F(y^{\Sigma F}),g^H(y^{\Sigma F})) \cup W(y^{\Sigma F})$	useless
9	(2.1b)	(1.1a),(0.9)	$R(f^F(y^{\Sigma G}),g^H(y^{\Sigma G})) \cup W(y^{\Sigma G})$	useless
10	(2.1c)	(1.1a),(0.9)	$R(f^F(y^{\Sigma H}),g^H(y^{\Sigma H})) \cup W(y^{\Sigma H})$	useless
11	(2.1d)	(1.1b),(0.9)	$R(f^F(y^{\Sigma E}),g^H(y^{\Sigma E})) \cup W(y^{\Sigma E})$	
12	(2.2a)	(1.1a),(0.10)	$Q(y^{\Sigma D},x^{\Sigma F}) \cup W(y^{\Sigma D})$	useless
13	(2.2b)	(1.1a),(0.10)	$Q(y^{\Sigma D},x^{\Sigma G}) \cup W(y^{\Sigma D})$	useless
14	(2.2c)	(1.1a),(0.10)	$Q(y^{\Sigma D},x^{\Sigma H}) \cup W(y^{\Sigma D})$	useless
15	(2.2d)	(1.1b),(0.10)	$Q(y^{\Sigma E},x^{\Sigma F}) \cup W(y^{\Sigma E})$	useless
16	(2.2e)	(1.1b),(0.10)	$Q(y^{\Sigma E},x^{\Sigma G}) \cup W(y^{\Sigma E})$	useless
17	(2.2f)	(1.1b),(0.10)	$Q(y^{\Sigma E},x^{\Sigma H}) \cup W(y^{\Sigma E})$	
		(1.1a),(0.11)	not resolvable	
18	(2.3)	(1.1b),(0.11)	$Q(x^{\Sigma I},x^{\Sigma E}) \cup R(f^F(x^{\Sigma I}),x^{\Sigma E})$	
19	(2.4)	(1.2),(0.9)	same as (2.1d)	
20	(2.5)	(1.2),(0.10)	$P(y^{\Sigma E}) \cup W(x^{\Sigma E})$	
21	(2.6)	(1.2),(0.11)	$P(y^{\Sigma I}) \cup R(f^F(y^{\Sigma I}),g^H(y^{\Sigma I}))$	
22	(2.7a)	(1.3a),(0.8)	same as (2.2a)	
23	(2.7b)	(1.3a),(0.8)	same as (2.2b)	
24	(2.7c)	(1.3b),(0.8)	same as (2.2c)	
25	(2.7d)	(1.3b),(0.8)	same as (2.2d)	
26	(2.7e)	(1.3c),(0.8)	same as (2.2e)	
27	(2.7f)	(1.3c),(0.8)	same as (2.2f)	
		(1.3a),(0.9)	not resolvable	
		(1.3b),(0.9)	not resolvable	
28	(2.8)	(1.3c),(0.9)	same as (2.5)	
29	(2.9a)	(1.3a),(0.11)	$P(x^{\Sigma I}) \cup Q(x^{\Sigma I},x^{\Sigma F})$	useless
30	(2.9b)	(1.3b),(0.11)	$P(x^{\Sigma I}) \cup Q(x^{\Sigma I},x^{\Sigma G})$	useless
31	(2.9c)	(1.3c),(0.11)	$P(x^{\Sigma I}) \cup Q(x^{\Sigma I},x^{\Sigma H})$	
32	(2.10)	(1.4),(0.8)	same as (2.3)	
33	(2.11)	(1.4),(0.9)	same as (2.6)	
34	(2.12a)	(1.4),(0.10)	same as (2.9a)	
35	(2.12b)	(1.4),(0.10)	same as (2.9a)	
36	(2.12c)	(1.4),(0.10)	same as (2.9a)	

–

37	(3.1a)	(2.1a),(0.10)	$W(y^{\Sigma F})$	useless
38	(3.1b)	(2.1b),(0.10)	$W(y^{\Sigma G})$	useless
39	(3.1c)	(2.1c),(0.10)	$W(y^{\Sigma H})$	useless
40	(3.1d)	(2.1d),(0.10)	$W(y^{\Sigma E})$	
		(2.1a),(0.11)	not resolvable	
		(2.1b),(0.11)	not resolvable	
		(2.1c),(0.11)	not resolvable	
41	(3.2)	(2.1d),(0.11)	$R(f^F(y^{\Sigma I}),g^H(y^{\Sigma I}))$	
		(2.2a),(0.9)	not resolvable	
		(2.2b),(0.9)	not resolvable	
42	(3.3a)	(2.2c),(0.9)	same as (3.1a)	
43	(3.3b)	(2.2c),(0.9)	same as (3.1b)	
44	(3.3c)	(2.2c),(0.9)	same as (3.1c)	
		(2.2d),(0.9)	not resolvable	
		(2.2e),(0.9)	not resolvable	
45	(3.3d)	(2.2f),(0.9)	same as (3.1d)	
		(2.2a),(0.11)	not resolvable	
		(2.2b),(0.11)	not resolvable	
		(2.2c),(0.11)	not resolvable	
46	(3.4a)	(2.2d),(0.11)	$Q(x^{\Sigma I},x^{\Sigma F})$	useless
47	(3.4b)	(2.2e),(0.11)	$Q(x^{\Sigma I},x^{\Sigma G})$	useless
48	(3.4c)	(2.2f),(0.11)	$Q(x^{\Sigma I},x^{\Sigma H})$	
49	(3.5)	(2.3),(0.9)	same as (3.2d)	
50	(3.6a)	(2.3),(0.10)	same as (3.4a)	
51	(3.6b)	(2.3),(0.10)	same as (3.4b)	
52	(3.6c)	(2.3),(0.10)	same as (3.4c)	
53	(3.7)	(2.5),(0.8)	same as (3.1d)	
54	(3.8)	(2.5),(0.11)	$P(y^{\Sigma I})$	
55	(3.9)	(2.6),(0.8)	same as (3.2)	
56	(3.10)	(2.6),(0.10)	same as (3.8)	
57	(3.11a)	(2.9a),(0.8)	same as (3.4a)	
58	(3.11b)	(2.9b),(0.8)	same as (3.4b)	
59	(3.11c)	(2.9c),(0.8)	same as (3.4c)	
		(2.9a),(0.9)	not resolvable	
		(2.9b),(0.9)	not resolvable	
60	(3.12)	(2.9c),(0.9)	same as (3.8)	
		(3.1a),(0.11)	not resolvable	
		(3.1b),(0.11)	not resolvable	
		(3.1c),(0.11)	not resolvable	
61	(4.1)	(3.1d),(0.11)	$\square$	
	(4.2a)	(3.2),(0.10)	$\square$	
		(3.3a),(0.10)	not resolvable	
		(3.3b),(0.10)	not resolvable	
		(3.3c),(0.10)	not resolvable	

(4.2b)	(3.3d),(0.11)	□
	(3.4a),(0.9)	not resolvable
	(3.4b),(0.9)	not resolvable
(4.3)	(3.4c),(0.9)	□
(4.4)	(3.8),(0.8)	□

APPENDIX D

Alternative Approaches of $R_W(\cdot)$

In Part II, it has been shown how a pair of variables satisfying a certain condition can be unified over the weakest range in *a Σ-extended* L_Σ^1. Such idea of unifying a pair of variables satisfying a certain condition can also be embodied by alternative approaches. They include: (i) an approach in which the theory in a many-sorted language L_m is repeatedly translated into *a revised language* L_m' of L_m along the way the refutation of the theory is carried out, and (ii) an approach in which the theory to be refuted is expressed in a generalized version (L_m^g) of an ordinary many-sorted language (L_m) whose variable sets and constant sets are not necessarily disjoint. These two alternative approaches are described in this appendix.

D.1. Refutation in a Revised Language L_m' of L_m

The first alternative approach is described. A many-sorted theory T_m^0 in a many-sorted language L_m^0 is considered. Let T_m^0 be the theory to be refuted. Let two clauses $\psi_a^0, \psi_b^0 \in T_m^0$ contain two variables v_i and v_j, respectively, and let these two clauses be resolvable if v_i and v_j are unified. When $|IM(Ran(v_i)) \cap IM(Ran(v_j))| > 1$, if the two clauses were expressed in L_Σ^1, L_Σ^1 can be extended to unify the two variables v_i and v_j over the weakest possible range, i.e., the intersection of the ranges of v_i and v_j. When the two clauses are expressed in L_m, however, unifying v_i and v_j over the weakest range is not allowed, although it becomes possible if the two clauses ψ_a^0 and ψ_b^0 are translated into a new language in which a variable ranging over the intersection of the ranges of

v_i and v_j can be introduced. That is, L_m^0 can be revised into another many-sorted language L_m^1 that is identical with L_m^0 except that in L_m^1 an additional variable set exists whose members range over the sort identical with the intersection of the ranges of v_i and v_j . Then T_m^0 can be translated into T_m^1 in L_m^1 including the translations of ψ_a^0 and ψ_b^0 into L_m^1 , say ψ_a^1 and ψ_b^1 , respectively. A resolvent, say ψ^1 , of ψ_a^1 and ψ_b^1 can be derived as a clause in L_m^1 . Let the overall process described so far be abbreviated by

$$T_m^1[T_m^0 \mid L_m^1] \vdash_{\overline{R(m)}} \psi^1$$

where $T_m^1[T_m^0 \mid L_m^1]$ means T_m^0 in L_m^0 is translated into T_m^1 by using the revised language L_m^1 of L_m^0 and the symbol " $\vdash_{\overline{R(m)}}$ " means the deduction process that derives ψ^1 in L_m^1 as a resolvent of a pair of member of $T_m^1[T_m^0 \mid L_m^1]$ (specifically, in the preceding example the pair would be ψ_a^1 and ψ_b^1).

The preceding example describes only a snapshot of the continuous process that is employed in this alternative approach. For example, ψ_a^1 and ψ^1 can also be resolved in a way similar to the one used in resolving ψ_a^0 and ψ_b^0 and if this is done, then resolving ψ_a^0 and ψ_b^0 means that a deduction process such as

$$T_m^2[T_m^1;\psi^1 \mid L_m^2] \vdash_{\overline{R(m)}} \psi^2$$

immediately follows the previous deduction process $T_m^1[T_m^0 \mid L_m^1] \vdash_{\overline{R(m)}} \psi^1$, where the symbol ";" is used to mean "in addition to". In summary, in this alternative approach (i) a theory and a logical consequence of the theory, i.e., a resolvent of a pair of clauses in the theory, are translated into a new language, (ii) another logical consequence is derived from the theory and the logical consequence in the new language, and (iii) the processes (i) and (ii) are repeated one after another until the intended logical consequence $\Box$ is derived.

In general, the refutation obtained in this approach can be viewed as a sequence of deduction processes of the following form:

$$T_m^1[T_m^0 \mid L_m^1] \vdash_{\overline{R(m)}} \psi^1$$
$$T_m^2[T_m^1;\psi^1 \mid L_m^2] \vdash_{\overline{R(m)}} \psi^2$$
$$\vdots$$
$$T_m^{p-1}[T_m^{p-2};\psi^{p-2} \mid L_m^{p-1}] \vdash_{\overline{R(m)}} \psi^{p-1}$$
$$T_m^p[T_m^{p-1};\psi^{p-1} \mid L_m^p] \vdash_{\overline{R(m)}} \Box$$

The preceding sequence of deduction processes makes it clear that in this alternative approach the inconsistency of the theory T_m^0 is not proved in L_m^0 but in L_m^p after various stages of theory translations in newly revised languages have been made. It must be justified whether the inconsistency proof of T_m^p made in L_m^p can be carried over to the inconsistency proof of the original theory T_m^0 in L_m^0. Since refuting a theory by this alternative approach entails a series of theory translation in a new language and derivation of a logical consequence from the translated theory, the following theorem can be shown first:

<u>Theorem D.1</u>

If $T_m^i[T_m^{i-1};\psi^{i-1} \mid L_m^i] \vdash_{\overline{R(m)}} \psi^i$, $i > 0$, then there is a translation of ψ^i into L_m^{i-1}, denoted by $\psi_i^i[\psi^i \mid L_m^{i-1}]$, and for the translation $\psi_i^i[\psi^i \mid L_m^{i-1}]$ there is a proof procedure " $\vdash_{\overline{R(?)}}$ " such that

$$T_m^{i-1};\psi^{i-1} \vdash_{\overline{R(?)}} \psi_i^i[\psi^i \mid L_m^{i-1}]$$

where $\psi^0 = \phi$.

In the preceding theorem, the existence of " $\vdash_{\overline{R(?)}}$ " means that $\psi_s^i[\psi^i \mid L_m^{i-1}]$ is a logical consequence of $T_m^{i-1};\psi^{i-1}$ although $\psi_s^i[\psi^i \mid L_m^{i-1}]$ is obtained indirectly via " $\vdash_{\overline{R(m)}}$ " and the translation of ψ^i into L_m^{i-1} . The following theorem must further be shown:

Theorem D.2

When $T_m^i[T_m^{i-1};\psi^{i-1} \mid L_m^i] \vdash_{\overline{R(m)}} \psi^i$, $i > 0$, if ψ^i is $\Box$ then $\psi_s^i[\psi^i \mid L_m^{i-1}]$ is also $\Box$.

By combining Theorem D.1 with the preceding theorem, it is implied that even if $\Box$ is derived from $T_m^i[T_m^{i-1};\psi^{i-1} \mid L_m^i]$ in L_m^p , for some $p > 0$, $\Box$ is also a logical consequence of T_m^0 .

D.2. Refutation in a generalized version L_m^g of L_m

In this approach, the theory to be refuted is expressed in a many-sorted language (L_m^g) that is a more general version than an ordinary many-sorted language (L_m) such as mentioned in Section 9.3 or given in [Ende72, KrKr67]. L_m^g is more general than L_m in the sense that in L_m^g neither its variable sets nor its constant sets need to be disjoint.

The language L_m^g is first defined. A *many-sorted language L_m^g with sort index set I* consists of the following: (1) $|I|$ infinite sets $V^1, \cdots, V^{|I|}$ (not necessarily disjoint) where the elements of V^i , $1 \leq i \leq |I|$, are called variables of sort i ; (2) $|I|$ sets $C^1, \cdots, C^{|I|}$ where the elements of C^i , $1 \leq i \leq |I|$, are called constant symbols of sort i such that

$C^{i_1} \cap \cdots \cap C^{i_n} \neq \phi$, $\{i_1, \cdots, i_n\} \subseteq I$, if and only if $V^{i_1} \cap \cdots \cap V^{i_n} \neq \phi$; (3) for each $n > 0$ and each n-tuple $<i_1, \cdots, i_n>$, $\{i_1, \cdots, i_n\} \subseteq I$, a set $R^{<i_1, \cdots, i_n>}$ whose elements are called relational symbols of sort $<i_1, \cdots, i_n>$; (4) for each $n > 0$ and each $n+1$-tuple $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, a set $F^{<i_1, \cdots, i_n, i_{n+1}>}$ whose elements are called function symbols of sort $<i_1, \cdots, i_n, i_{n+1}>$; (5) logical connectives $\neg$ and $\rightarrow$; and (6) a universal quantifier $\forall$.

Definable symbols $\cup, \cap, \leftrightarrows$ and $\exists$ are introduced in L_m^g as usual and the syntax rule of L_m^g is also given as usual. The interpretation of the formulas of L_m^g with sort index set I is given as follows. Let $MS(L_m^g)$ stand for a many-sorted structure associated with L_m^g. $MS(L_m^g)$ consists of: (1) $|I|$ nonempty sets of objects $S_1, \cdots, S_{|I|}$ where S_i is called the domain of sort i of $MS(L_m^g)$ such that $S_{i_1} \cap \cdots \cap S_{i_n} \neq \phi$, $\{i_1, \cdots, i_n\} \subseteq I$, if and only if $V^{i_1} \cap \cdots \cap V^{i_n} \neq \phi$; (2) for each constant symbol $c \in C^{i_1} \cap \cdots \cap C^{i_n}$, $\{i_1, \cdots, i_n\} \subseteq I$, an element $c^{MS} \in S_{i_1} \cap \cdots \cap S_{i_n}$; (3) for each predicate symbol P of sort $<i_1, \cdots, i_n>$, a relation $P^{MS} \subseteq S_{i_1} \times \cdots \times S_{i_n}$; (4) for each function symbol f of sort $<i_1, \cdots, i_n, i_{n+1}>$, a function $f^{MS} : S_{i_1} \times \cdots \times S_{i_n} \rightarrow S_{i_{n+1}}$.

A variable assignment function s is given as follows: If $V = \bigcup_{i \in I} V^i$ where V^i is a variable set of L_m, then s is an assignment function, $s : V \rightarrow \bigcup_{i \in I} S_i$, such that for a variable $x_i \in V^{i_1} \cap \cdots \cap V^{i_n}$, $\{i_1, \cdots, i_n\} \subseteq I$, $s(x_i) = a$, where $a \in S_{i_1} \cap \cdots \cap S_{i_n}$. Assignment function for the terms of L_m^g is defined as usual. The validity of each formula in $MS(L_m^g)$ is determined as usual.

As long as L_m^g is a more general version than L_m , it trivially follows that any formula in L_m can be expressed in L_m^g . However, the converse must be shown to assure that L_m^g is as legitimate as L_m . The converse is shown in Appendix E.

It is shown how the second alternative approach can be carried out. Let a theory T_o in a one-sorted language (L_o) be equivalently expressed as a many-sorted theory, say T_m , in an ordinary many-sorted language L_m with sort index I . Let the language for T_m be $L_m(T_m)$. Let S_i and S_j , $i, j \in I$, be two unary predicate symbols in L_o which are defined correspondingly to sort i and j of $L_m(T_m)$. An inflexible usage of sort variables of L_m is displayed when another formula in L_o , say a logical consequence ϕ_o of T_o ,

$$\phi_o = \forall x \ \ (S_i(x) \cap S_j(x) \ \rightarrow \ \psi(x)) \tag{D.1}$$

needs to be further abbreviated in $L_m(T_m)$. The syntax of L_m does not allow the one-sorted expression ϕ_o to be abbreviated into a many-sorted expression that is compact enough to carry out the idea behind $R_W(\cdot)$, for instance, as compact as the form (D.2) below, unless $L_m(T_m)$ is appropriately revised to do so.

Let $L_m^g(T_m)$ be the L_m^g defined to be equivalent to $L_m(T_m)$, i.e., $L_m^g(T_m)$ is identical with $L_m(T_m)$ except that in $L_m^g(T_m)$ its variable sets and its constant sets do not need to be disjoint. Let a variable $x_{i,j}$ belong to sorts i and j , i.e., $x_{i,j} \in V_g^i$ and $x_{i,j} \in V_g^j$ where V_g^i and V_g^j are variable sets of sort i and j in $L_m^g(T_m)$. Then ϕ_o in (D.1) can be abbreviated into the form

$$\forall x_{i,j} \ \ \psi(x_{i,j}) \tag{D.2}$$

in $L_m^g(T_m)$. It is noticed that abbreviating the one-sorted expression of the form

(D.1) into a many-sorted expression of the form (D.2) is the only type of abbreviation needed when embodying the idea behind $R_W(\cdot)$. Therefore an alternative approach of $R_W(\cdot)$ is obtained by expressing the theory to be refuted in L_m^g.

APPENDIX E

Translation of a Formula in L_m^g into L_m

It is shown that any formula in the many-sorted language L_m^g that was defined in Appendix D can be translated in an ordinary many-sorted language L_m . Showing this implies that L_m^g is as legitimate as L_m which is commonly given in various literature such as [KrKr67] and [Ende72].

Given a L_m^g with sort index set I , its corresponding ordinary many-sorted L_m is constructed. A few preliminary steps are given first. Let $V = \{V^i : i \in I\}$ where V^i is a variable set of L_m^g . A set V_0 of disjoint variable sets is derived from V as follows: for each $k \in I$, if $U_k = \{V^k , \bigcup_{i \in I} V^i - V^k\}$ then $V_0 = \bigsqcap_{k \in I} U_k$ †. Let I_0 be the index set for V_0 so that each element of V_0 is expressed by V_0^k, $k \in I_0$. A relationship holds between V and V_0 : If ξ be a function $\xi : I \rightarrow N^+$ where N^+ is the positive integer set, then for each $V^i \in V$ there exist uniquely $\xi(i)$ variable sets in V_0 such that $V^i = V_0^{j_1} \cup \cdots \cup V_0^{j_{\xi(i)}}$, $\{j_1 , \cdots , j_{\xi(i)}\} \subseteq I_0$. In the preceding relationship, $\{j_1 , \cdots , j_{\xi(i)}\} \subseteq I_0$ is said to be the i's, $i \in I$, corresponding index subset, denoted by $CI(i)$, of I_0 .

The ordinary many-sorted language L_m with the sort index set I_0 that corresponds to the L_m^g then consists of: (1) $|I_0|$ infinite disjoint variable sets $V_0^1 , \cdots , V_0^{|I_0|}$ such that $\{V_0^i : i \in I_0\} = V_0$; (2) $|I_0|$ disjoint constant sets

† The definition of " $\bigsqcap$ " was given at Section 7.3 as follows: For two partitions Π^1 and Π^2 of a set, $\Pi^1 \bigsqcap \Pi^2 = \{ S : S = B_i \cap B_j$ where $B_i \in \Pi^1$ and $B_j \in \Pi^2$, and $S \neq \phi \}$. Since the commutativity and the associativity hold for $\bigsqcap$, let $\Pi^1 \bigsqcap \cdots \bigsqcap \Pi^n$ be written by $\bigsqcap_{i \in \{1, \cdots, n\}} \Pi^i$.

$C_o^1, \cdots, C_o^{|I_o|}$ such that to each C^i, $i \in I$, of L_m^g if $CI(i) = \{j_1, \cdots, j_{\xi(i)}\}$ then $C^i = C_o^{j_1} \cup \cdots \cup C_o^{j_{\xi(i)}}$; (3) to each predicate symbol P of sort $<i_1, \cdots, i_n>$, $\{i_1, \cdots, i_n\} \subseteq I$, of L_m^g its corresponding $\xi(i_1) \times \cdots \times \xi(i_n)$ different relation symbols, whose collection is denoted by $CP(P)$, such that for each k, $1 \le k \le \xi(i_1) \times \cdots \times \xi(i_n)$, $P_k \in CP(P)$ is a relation symbol of sort $<i_1^k, \cdots, i_n^k> \in CI(i_1) \times \cdots \times CI(i_n)$, and; (4) to each function symbol f of sort $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n\} \subseteq I$, of L_m^g its corresponding $\xi(i_1) \times \cdots \times \xi(i_{n+1})$ different function symbols, whose collection is denoted by $CF(f)$, such that for each k, $1 \le k \le \xi(i_1) \times \cdots \times \xi(i_{n+1})$, $f_k \in CF(f)$ is a function symbol of sort $<i_1^k, \cdots, i_n^k, i_{n+1}^k> \in CI(i_1) \times \cdots \times CI(i_{n+1})$.

A few notation are introduced. Let $TERM(L_m^g)$ and $TERM(L_m)$ be the sets of all the terms of L_m^g and L_m, respectively. For each $t \in TERM(L_m^g)$ *its corresponding terms in* $TERM(L_m)$, whose collection is denoted by $CT(t)$, is defined inductively as follows: (i) if t is a variable $x \in V^i$, $i \in I$, and $CI(i) = \{j_1, \cdots, j_{\xi(i)}\}$, then $CT(t) = \{x_{j_1}^o, \cdots, x_{j_{\xi(i)}}^o\}$ where for each $k \in \{j_1, \cdots, j_{\xi(i)}\}$, $x_k^o \in V_o^k$; (ii) if t is a constant c of sort i, $i \in I$, and $CI(i) = \{j_1, \cdots, j_{\xi(i)}\}$, then $CT(t) = \{c_{j_1}^o, \cdots, c_{j_{\xi(i)}}^o\}$ where for each $k \in \{j_1, \cdots, j_{\xi(i)}\}$, $c_k^o \in C_o^k$; (iii) if t is a term of the form $f(t_1, \cdots, t_n)$ where f is an n-place function symbol of sort $<i_1, \cdots, i_n, i_{n+1}>$, $\{i_1, \cdots, i_n, i_{n+1}\} \subseteq I$, then $CT(t) = \{f_k^o(t_1^o, \cdots, t_n^o) : f_k^o \in CF(f)$ and $t_j^o \in CT(t_j), 1 \le j \le n\}$.

Now let *the terms of a formula,* say ψ, be defined as follows: (i) if $P(t_{i_1}, \cdots, t_{i_m})$ is an atomic subformula of ψ where P is an m-place relation symbol and $t_{i_1}, \cdots, t_{i_m}$ are terms, then $t_{i_1}, \cdots, t_{i_m}$ are terms of ψ, and (ii) no terms other than those identified by (i) are terms of ψ. When it is convenient, ψ is expressed by $\psi[t_1, \cdots, t_n]$ if $t_1, \cdots, t_n$ are all the terms of ψ in the order of their appearance in ψ [notice that there can be duplicate terms among $t_1, \cdots, t_n$].

It is shown how a formula in L_m^g with the sort index set I is translated into the L_m with the sort index set I_o which was constructed correspondingly to the L_m^g. Let σ_m be a formula in L_m^g with the sort index set I. Let σ_m be of the form,

$$\sigma_m[t_1, \cdots, t_l]. \tag{b.1}$$

Let $k = |CT(t_1)| \times \cdots \times |CT(t_l)|$. The translation of σ_m into σ_m^o in L_m with the sort index I_o is then

$$\sigma_m^o = \bigcup_{1 \leq j \leq k} \sigma_m^j[t_1^o(j), \cdots, t_l^o(j)] \tag{b.2}$$

where for each j, $1 \leq j \leq k$, $\sigma_m^j[t_1^o(j), \cdots, t_l^o(j)]$ is constructed in the following way: (i) $t_1, \cdots, t_l$ of σ_m are replaced by $t_1^o(j), \cdots, t_l^o(j)$, respectively, where $<t_1^o(j), \cdots, t_l^o(j)>$ is the j^{th} element of $CT(t_1) \times \cdots \times CT(t_l)$; (ii) if $P(t_{u_1}, \cdots, t_{u_r})$, $\{u_1, \cdots, u_r\} \subseteq \{1, \cdots, l\}$, is an atomic subformula of σ_m such that by the step (i) the terms $t_{u_1}, \cdots, t_{u_r}$ are replaced by $t_{u_1}^o(j), \cdots, t_{u_r}^o(j)$, respectively, then P is replaced by $P^o \in CP(P)$ of sort $<j_1, \cdots, j_r>$, $\{j_1, \cdots, j_r\} \in I_o$, where each j_h, $1 \leq h \leq r$, is the sort to which $t_{u_h}^o(j)$ belongs, and; (iii) if Qx where Q is either $\forall$ or $\exists$ is a quantifier appeared in σ_m and by step (ii) the variable x appeared in $t_1, \cdots, t_l$ of σ_m is replaced by x^o, then Qx is replaced by Qx^o in σ_m^j.

As far as semantics for the formula σ_m^o of (b.2) is concerned, the structure for L_m , say $MS^o(L_m)$, can be constructed from the many-sorted structure for L_m^g, say $MS(L_m^g)$. Let $MS(L_m^g)$ be a quadratuple $MS(L_m^g) = < \{S_i\}_{i \in I} , R , F , C >$ where I is the sort index set, $\{S_i\}_{i \in I}$, the sorts of $MS(L_m^g)$, R , the relation set, F , the function set and C , the constant set. Then $MS^o(L_m)$ consists of: (1) $|I_o|$ nonempty disjoint sets of objects $S_1^o , \cdots , S_{|I_o|}^o$ such that for each S_i , $i \in I$, if $CI(i) = \{j_1 , \cdots , j_{\xi(i)}\}$, then $S_i = S_{j_1}^o \cup \cdots \cup S_{j_{\xi(i)}}^o$; (2) for each constant symbol c of sort i^k , $i^k \in I_o$, an element $c^{MS^o(L_m)}$ such that $c^{MS^o(L_m)} = c^{MS(L_m^g)}$, where $c^{MS(L_m^g)} \in C$; (3) for each predicate symbol P_k^o in L_m^g of sort $<i_1^k , \cdots , i_n^k>$, $\{i_1^k , \cdots , i_n^k\} \subseteq I_o$, if $P_k^o \in CP(P)$, then the relation $P_k^{MS^o(L_m)} = P^{MS(L_m^g)} \cap (S_{i_1^k}^o \times \cdots \times S_{i_n^k}^o)$, where $P^{MS(L_m^g)} \in R$; (4) for each function symbol f_k^o in L_m of sort $<i_1^k , \cdots , i_n^k , i_{n+1}^k >$, $\{i_1^k , \cdots , i_n^k\} \subseteq I_o$, if $f_k^o \in CF(f)$, then the function $f_k^{MS^o(L_m)} : S_{i_1^k}^o \times \cdots \times S_{i_n^k}^o \rightarrow S_{i_{n+1}^k}^o$ such that $f_k^o(a_{i_1}^k , \cdots , a_{i_n}^k) = f(a_{i_1}^k , \cdots , a_{i_n}^k)$, where $f^{MS(L_m^g)} \in F$.

Finally, the following theorem is shown for the translation of the formula σ_m of (b.1) into the formula σ_m^o of (b.2):

<u>Theorem B.1</u>

A sentence σ_m in L_m^g is true in $MS(L_m^g)$ iff σ_m^o in L_m is true in $MS^o(L_m)$.

Proof. Proof is by induction on the length of σ_m . First, proof is given for when σ_m is atomic. Let σ_m be an atomic formula of the form $R(t_1 , \cdots , t_n)$ where R is an n-place relation symbol of sort $<i_1 , \cdots , i_n >$, $\{i_1 , \cdots , i_n\} \in I$, and

t_i's are terms. Let σ_m be true in $MS(L_m^*)$ with an assignment function s. The translation of σ_m into σ_m^* in L_m is the following:

$$\sigma_m^o = R_1^o(t_1^o(1), \cdots, t_n^o(1)) \cup \cdots \cup R_k^o(t_1^o(k), \cdots, t_n^o(k))$$

where $k = |CT(t_1)| \times \cdots \times |CT(t_n)|$ and for each j, $1 \leq j \leq k$, $R_j^o \in CP(R)$ and $<t_1^o(j), \cdots, t_n^o(j)> \in CT(t_1) \times \cdots \times CT(t_n)$.

Along with the preceding translation, an assignment function s^o for the variables of L_m is introduced as follows: s^o is an assignment function $s^o : \bigcup_{i \in I_o} V_o^i \rightarrow \bigcup_{i \in I_o} S_i^o$ such that if x in σ_m is replaced by x^o during the translation of σ_m into σ_m^o, then $s(x) = s^o(x^o)$ [the assignment function s^o defined here is used throughout this proof]. For notational convenience the symbol s^o is also used for the assignment for terms. It can be trivially shown that for a unique j, $1 \leq j \leq k$,

$$<s(t_1), \cdots, s(t_n)> = <s^o(t_1^o(j)), \cdots, s^o(t_n^o(j))> \cdots (1).$$

Let L be the index set for $CP(R)$ so that each member of $CP(R)$ is expressed by R_l, $l \in L$. Then from the way that $MS^o(L_m)$ is defined, it follows that

$$R^{MS(L_m^*)} = \bigcup_{l \in L} R_l^{MS^o(L_m)} \cdots (2).$$

From the way σ_m^o is constructed, it follows that

$$\bigcup_{l \in L} R_l^{MS^o(L_m)} = \bigcup_{1 \leq j \leq k} R_j^{MS^o(L_m)} \cdots (3)$$

where each R_j^o, $1 \leq j \leq k$, is an atomic subformula in σ_m^o. From (1) the following also holds: For each j_h, $1 \leq j_h \leq k$, if for some j^* $1 \leq j^* \leq k$,

$<s^o(t_1^o(j_h)), \cdots, s^o(t_n^o(j_h))> \in R_{j^*}^{MS^o(L_m)}$ then for any j', $j' \neq j^*$ and $1 \leq j' \leq k$,

$$<s^o(t_1^o(j_h)), \cdots, s^o(t_n^o(j_h))> \notin R_{j'}^{MS^o(L_m)} \quad \cdots (4)$$

Consequently, the following holds:

$\models_{MS(L_m^f)} R(t_1, \cdots, t_n)[s]$

$\Longleftrightarrow \; <s(t_1), \cdots, s(t_n)> \in R^{MS(L_m^f)}$

$\Longleftrightarrow$ for some j, $1 \leq j \leq k$, $<s^o(t_1^o(j)), \cdots, s^o(t_n^o(j))> \in R^{MS(L_m^f)}$ from (1)

$\Longleftrightarrow$ for some j, $1 \leq j \leq k$, $<s^o(t_1^o(j)), \cdots, s^o(t_n^o(j))> \in \bigcup_{l \in L} R_l^{MS^o(L_m)}$ from (2)

$\Longleftrightarrow$ for some j, $1 \leq j \leq k$, $<s^o(t_1^o(j)), \cdots, s^o(t_n^o(j))> \in \bigcup_{1 \leq j \leq k} R_j^{MS^o(L_m)}$ from (3)

$\Longleftrightarrow \; \bigcup_{1 \leq j \leq k} \{$ for some j', $1 \leq j' \leq k$, $<s^o(t_1^o(j')), \cdots, s^o(t_n^o(j'))> \in R_j^{MS^o(L_m)} \}$

$\Longleftrightarrow \; \bigcup_{1 \leq j \leq k} \{<s^o(t_1^o(j)), \cdots, s^o(t_n^o(j))> \in R_j^{MS^o(L_m)}\}$ from (4)

$\Longleftrightarrow \; \bigcup_{1 \leq j \leq k} \{ \models_{MS^o(L_m)} R_j^o(t_1^o(j), \cdots, t_n^o(j))[s^o] \}$

$\Longleftrightarrow \; \models_{MS^o(L_m)} \bigcup_{1 \leq j \leq k} R_j^o(t_1^o(j), \cdots, t_n^o(j))\ [s^o]$.

It is concluded that when σ_m is atomic, $\models_{MS(L_m^f)} \sigma_m$ iff $\models_{MS^o(L_m)} \sigma_m^o$.

Suppose that the result is true for all formulas of length less than or equal to h. It is shown that the inductive step holds for the formulas of length $h+1$. Let $\sigma_{m,1}$ and $\sigma_{m,2}$ be formulas of length h, and for $\sigma_{m,1}^o$ and $\sigma_{m,2}^o$ it holds that $\models_{MS(L_m^f)} \sigma_{m,1}$ iff $\models_{MS^o(L_m)} \sigma_{m,1}^o$ and $\models_{MS(L_m^f)} \sigma_{m,2}$ iff $\models_{MS^o(L_m)} \sigma_{m,2}^o$. Inductive step is the following:

(Case I) Let σ_m be $\neg \sigma_{m,1}$. It is trivial to show that

$$\models_{MS(L_m^I)} \sigma_m [s] \iff \models_{MS^o(L_m)} \neg \sigma_{m,1}^o [s^o] .$$

(Case II) Let σ_m be $\sigma_{m,1} \to \sigma_{m,2}$. It is trivial to show that

$$\models_{MS(L_m^I)} \sigma_m [s] \iff \models_{MS^o(L_m)} \sigma_{m,2}^o \to \sigma_{m,2}^o [s^o] .$$

(Case III) Let σ_m be $\forall x_i \sigma_{m,1}$ where $x_i \in V^i$. The followings holds: If $CI(i) = \{i_1, \cdots, i_{\xi(i)}\}$, then

$$S_i = S_{i_1}^o \cup \cdots \cup S_{i_{\xi(i)}}^o \quad \cdots (5)$$

and for any $i_n, i_m \in \{i_1, \cdots, i_{\xi(i)}\}$

$$S_{i_n}^o \cap S_{i_m}^o = \phi \quad \cdots (6) .$$

Let $t_1, \cdots, t_n$ be the terms of $\sigma_{m,1}$ and let $\sigma_{m,1}^o = \bigcup_{1 \le j \le k} \sigma_{m,1}^j [t_1^o(j), \cdots, t_n^o(j)]$ where $k = |CT(t_1)| \times \cdots \times |CT(t_n)|$. From the induction hypothesis, it follows that: for each j, $1 \le j \le k$, if $x_i^o(j) \in CT(x_i)$, $x_i^o(j) \in V_o^{i_l}$, $i_l \in \{i_1, \cdots, i_{\xi(i)}\}$, and $a^o(j) \in S_{i_l}^o$, then

$$\models_{MS(L_m^I)} \sigma_{m,1} [s(x_i \mid a)] \iff \models_{MS^o(L_m)} \bigcup_{1 \le j \le k} \sigma_{m,1}^j \; [s^o(x_i^o(j) \mid a^o(j))] \quad \cdots (7) .$$

Let η be a function $\eta : \{1, \cdots, k\} \to \{i_1, \cdots, i_{\xi(i)}\}$ such that for each $a^o(j)$, $1 \le j \le k$, in (8), if $a^o(j) \in S_{i_l}^o$ then $\eta(j) = l$. Finally, the following holds:

$$\models_{MS(L_m^g)} \sigma_m \, [s] \iff \models_{MS(L_m^g)} \forall x_i \; \sigma_{m,1} \, [s]$$

$$\iff \text{for any } a \in S_i \, , \; \models_{MS(L_m^g)} \sigma_{m,1} \, [s(x_i \mid a)]$$

$$\iff \text{for any } a \in S_{i_1}^o \cup \cdots \cup S_{i_{g(i)}}^o \; \models_{MS(L_m^g)} \sigma_{m,1} \, [s(x_i \mid a)]$$

from (5)

$$\iff \bigcup_{1 \le j \le k} \{\text{for any } a^o(j) \in S_{i_{\eta(j)}}^o , \; \models_{MS^o(L_m)} \sigma_{m,1}^j \, [s^o(x_i^o(j) \mid a^o(j))]\}$$

from (6) and (7)

$$\iff \bigcup_{1 \le j \le k} \{ \models_{MS^o(L_m)} \forall x_i^o(j) \, \sigma_{m,1}^j \, [s^o] \}$$

$$\iff \models_{MS^o(L_m)} \bigcup_{1 \le j \le k} \forall x_i^o(j) \; \sigma_{m,1}^j \; [s^o] \, .$$

Since $\bigcup_{1 \le j \le k} \forall x_i^o(j) \, \sigma_{m,1}^j$ is σ_m^o, it follows that $\models_{MS(L_m^g)} \sigma_m$ iff $\models_{MS^o(L_m)} \sigma_m^o$.

Q.E.D.

BIBLIOGRAPHY

BIBLIOGRAPHY

[AnBl70] Anderson, R. and Bledsoe, W. W., "A linear format for resolution with merging and a new technique for establishing completeness," *J. ACM,* Vol. 17, pp. 525-534, 1970.

[AnBo73] Anderson, J. and Bower, G., *Human Associative Memory,* Winston, Washington D.C., 1973.

[Andr81] Andrew, P. B., "Theorem proving via general matings," *J. ACM,* Vol. 28, No. 2, pp. 193-214, April 1981.

[Aper81] Apers, P. M. G., "Redundant allocation of relations in a communication network," *Proc. Berkeley Workshop on Distributed Data Management and Computer Networks,* Lawrence Berkeley Lab., The University of California, Berkeley, 1981.

[BaFe81] Barr, A. and Feigenbaum, E. A., *The Handbook of Artificial Intelligence,* Vol. 1, 2, and 3, William Kaufmann, Inc., Los Altos, CA, 1981.

[Bern76] Bernstein, P. A., "Synthesizing third normal form relations from functional dependencies," *ACM Transactions on Database Systems,* Vol. 1, No. 4, Dec. 1976.

[Bern81] Bernstein, P. A. et. al., "Query processing in a system for distributed databases(SDD-1)," *ACM Transactions on Database Systems,* Vol. 6, No. 4, pp. 602-625, March 1981.

[Bibe81] Bibel, W., "On matrices with connections," *J. ACM,* Vol. 28, No. 4, pp. 633-645, Oct. 1981.

[BoWi77] Bobrow, D. G. and Winograd, T., "An overview of KRL, a knowledge representation language," *Cognitive Science,* Vol. 1, pp. 3-36, 1977.

[Boye71] Boyer, R. S., "Locking: A restriction of resolution," Ph. D. Dissertation, The University of Texas, Austin, 1971.

[Brac78] Brachman, R. J., "A structural paradigm for representing knowledge," Rep. No. 3605, Bolt Beranek and Newman, Inc., Cambridge, MA, 1978.

[BuFe78] Buchanan, B. G. and Feigenbaum. E. A., "DENDRAL and Meta-DENDRAL: Their applications dimension," *Artificial Intelligence,* Vol. 11, pp. 5-24, 1978.

[BuHa79] Buckles, B. P. and Hardin, D. M., "Partitioning and allocation of logical resources in a distributed computing environment," *Distributed System Design,* Tutorial IEEE Catalog No. EHO261-1, 1979.

[Carb70] Carbonell, J. R., "AI in CAI: An artificial intelligence approach to computer-assisted instruction," *IEEE Transactions on Man-Machine Systems,* Vol. MMS-11, pp. 190-202, 1970.

[Case72] Casey, R. G., "Allocation of copies of a file in an information network," *Proc. AFIPS Spring Joint Comput. Conf.,* Vol. 40, AFIPS Press, Arlington, VA, pp. 617-625, 1972.

[CeNW83] Ceri, S., Navathe, S. and Wiederhold, G., "Distributed design of logical database schemas," *IEEE Transactions on Software Engineering, Vol. SE-9, No. 4,* pp. 487-504, July 1983.

[Cham78] Champeaux, D. de, "A theorem prover dating a semantic network," *Proc. AISB/GI Conf.,* Hamburg, West Germany, 1978.

[Chan70] Chang, C. L., "The unit proof and the input proof in theorem proving," *J. ACM,* Vol. 17, No. 4, pp. 698-707, Oct. 1970.

[Chan76] Chang, C. L., "DEDUCE -- A deductive query language for relational data bases," *Pattern Recognition and AI,* Academic Press, New York, pp. 108-134, 1976.

[Chan78] Chang, C. L., "DEDUCE2: Further investigations of deduction in relational database," *Logic and Databases* (H. Gallaire and J. Minker, eds), Plenum Press, New York, pp. 201-236, 1978.

[Chen76] Chen, P., "The entity-relationship model -- toward a unified view of data," *ACM Transactions on Database Systems,* Vol. 1, No. 1, pp. 9-36, March 1976.

[Chu 69] Chu, W. W., "Optimal file allocation in a multiple computer network," *IEEE Transactions on Computer,* Vol. C-18, No. 10, pp. 885-889, Oct. 1969.

[Chu 73] Chu, W. W., "Optimal file allocation in a computer network," *Computer-Communication Network* (N. Abramson and F. Kuc, eds), Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

[Chu 76] Chu, W. W., "Performance of file directory systems for databases in star and distributed networks," *Proc. AFIPS Conf. NCC,* Vol. 45, AFIPS Press, Montvale, NJ, pp. 577-587, 1976.

[Chun83] Chung, C-W., *A Query Optimization for Distributed Database Systems,* Ph. D. Dissertation, The University of Michigan, Ann Arbor, MI, 1983.

[Codd70] Codd, E. F., "A relational model of data for large shared data banks," *Comm. ACM,* Vol. 13, June 1970.

[Codd72a] Codd, E. F., "Further normalization of the database relational model," *Database System* (R. Rustin, ed), Prentice-Hall, Inc., Englewood Cliffs, NJ, pp. 33-64, 1972.

[Codd72b] Codd, E. F., "Relational completeness of database sublanguages," *Database System* (R. Rustin, ed), Prentice-Hall, Inc., Englewood Cliffs, NJ, pp. 65-98, 1972.

[Codd78] Codd, E. F., "How about recently?," *Databases: Improving usability and responsiveness* (B. Shneiderman, ed), Academic Press, New York, pp. 3-28, 1978.

[Codd79] Codd, E. F., "Extending the database relational model to capture more meaning," *ACM Transactions on Database Systems,* Vol. 4, No. 4, pp. 397-434, 1979.

[CoGP81] Coffman, E. G. Jr., Gelenbe, E. and Plateau, B., "Optimization of the number of copies in a distributed data base," *IEEE Transactions on Software Engineering,* Vol. SE-7, No. 1, Jan. 1981.

[Cohn83] Cohn, A. G., "Improving the expressiveness of many sorted logic," *Proc. National Conf. on Artificial Intelligence,* pp. 84-87, 1983.

[DaPu60] Davis, M and Putnam, H., "A computing procedure for quantification theory," *J. ACM,* Vol. 7, pp. 201-215, March 1960.

[Date83] Date, C. J., *An Introduction to Database Systems,* Vol. 2, Addison-Wesley, Reading, MA, 1983.

[Davi72] Davis, D. J. M., "POPLER: A POP-2," Rep. No. MIP-89, School of AI, University of Edinburgh, Scotland, 1972.

[Delo78] Delobel, C., "Normalization and hierarchical dependencies in the relational data model," *ACM Transactions on Database Systems,* Vol. 3, No. 3, pp. 201-222, Sept. 1978.

[DePa82] De Bra, P. and Paredaens, J., "Horizontal decompositions and their impact on query solving," *SIGMOD Record,* Vol. 13, No. 1, pp. 46-50, Sept. 1982.

[DoFo82] Dowdy, L. W. and Foster, D. V., "Comparative models of the file assignment problem," *Computing Surveys,* Vol. 14, No. 2, pp. 287-313, June 1982.

[DuHN76] Duda, R. O., Hart, P. E. and Nilsson, N. J., "Subjective Bayesian methods for rule-based inference systems," *Proc. National Computer Conf.* (AFIPS Conf. Proc.), Vol. 45, pp. 1075-1082, 1976.

[Ende72] Enderton, H. B., *A Mathematical Introduction to Logic,* Academic Press, New York, 1972.

[EpSW78] Epstein, R., Stonebraker, M. and Wong, E., "Distributed query processing in a relational database systems," *Proc. ACM SIGMOD Int. Conf. on Management of Data,* pp. 169-180, June 1978.

[Eswa74] Eswaren, K. P., "Placement of records in a file and file allocation in a computer network," *Proc. IFIP Congress* (Information Processing 74), North-Holland, Amsterdam, 1974.

[FiHN72] Fikes, R. E., Hart, P., and Nilsson, N. J., "Learning and executing generalized robot plans," *Artificial Intelligence,* Vol. 3, pp. 251-288, 1972.

[FiHo80] Fisher, M. L. and Hochbaum, D. S., "Database location in computer networks," *J. ACM,* Vol. 27, No. 4, pp. 718-735, Oct. 1980.

[FiWe76] Filman, R. E. and Weyhrauch, R. W., "An FOL primer," Memo. 288, AI Laboratory, Stanford University, 1976.

[FuLa79] Fung, K. T. and Lam, C. M., "Optimal data allocation in a distributed database," *Proc. Trends and Applications,* National Bureau of Standards, Gaithesburg, Md., pp. 111-116, 1979.

[Furt81] Furtado, A. L., "Horizontal decomposition to improve a non-BCNF scheme," *SIGMOD Record,* Vol. 12, No. 1, pp. 26-32, Oct. 1981.

[GaMi78] Gallaire, H. and Minker, J. (eds), *Logic and Databases,* Plenum Press, New York, 1978.

[Gilm60] Gilmore, P. C., "A proof method for quantification theory: its justification and realization," *IBM J. Research Dev.,* Vol. 4, pp. 28-35, Jan. 1960.

[GoRo77] Goldstein, I. P. and Roberts, R. B., "NUDGE, a knowledge-based scheduling program," *Proc. Int. Joint Conf. on Artificial Intelligence,* Cambridge, MA, pp. 257-263, 1977.

[Gree69] Green, C. C., "The application of theorem-proving to question-answering systems," *Proc. Int. Joint Conf. on Artificial Intelligence,* Washington, D. C., pp. 219-237, 1969.

[Hail57] Hailperin, T., "A theory of restricted quantification I," *J. Symbolic Logic,* Vol. 22, No. 1, pp. 19-35, March 1957.

[HaMc78] Hammer, M. and McLeod, D., "The Semantic Model: A modelling mechanism for DB applications," *Proc. ACM SIGMOD Int. Conf. on Management of Data,* Austin, TX, May 1978.

[HaNi79] Hammer, M. and Niamir, B., "A heuristic approach to attribute partitioning," *Proc. ACM SIGMOD Int. Conf. on Management of Data,* pp. 93-101, May 1979.

[Haye71] Hayes, P., "A logic of actions," *Machine Intelligence,* Vol. 6, Metamathematics Unit, University of Edinburgh, 1971.

[Haye77] Hayes, P. J., "In defence of logic," *Proc. Int. Joint Conf. on Artificial Intelligence,* Cambridge, MA, pp. 559-565, 1977.

[HaZd80] Hammer, M. and Zdonik, S. B. Jr., "Knowledge-based query processing," *Proc. Int. Conf. on Very Large Data Bases,* Montreal, Canada, pp. 137-147, Oct. 1980.

[Hend75] Hendrix, G. G., "Expanding the utility of semantic networks through partitioning," *Proc. Int. Joint Conf. on Artificial Intelligence,* Tbilisi, Georgia, USSR, pp. 115-121, 1975.

[Hend78] Hendrix, G. G. et. al., "Developing a natural language interface to complex data," *ACM Transactions on Database Systems,* Vol. 3, No. 3, pp. 105-147, 1978.

[Hens72] Henschen, L. J., "N-sorted logic for automatic theorem proving in higher-order logic," *Proc. ACM Conference,* Boston, MA, 1972.

[Herb30] Herbrand, J., *Recherches sur la Theorie de la Demonstration (These Paris),* Warsaw (1930) Chapter 3, 1930. Also in *Logical Writings* (W. D. Goldfarb, ed.), D. Reidel Pub. Co., 1971.

[Hewi72] Hewitt, C., "Description and theoretical analysis (using schemata) of PLANNER, a language for proving theorems and manipulating models in a robot," Rep. No. TR-258, AI Laboratory, Massachusetts Institute of Technology, 1972.

[HeYa79] Hevner, A. R. and Yao, S. B., "Query processing in distributed database systems," *IEEE Transactions on Software Engineering,* Vol. SE-5, No. 3, pp. 177-187, May 1979.

[Horn51] Horn, A., "On sentences which are true of direct unions of algebra," *J. Symbolic Logic,* Vol. 16, pp. 14-21, 1951.

[Idel64] Idelson, A. V., "Calculi of constructive logic with subordinate variables," *American Mathematical Society Translations (2),* Vol. 99, 1972 -- translation of Trudy Mat. Inst. Steklov. 72, 1964.

[IrKh79] Irani, K. B. and Khabbaz, N. G., "A model for combined communication network design and file allocation for distributed databases," *Proc. Int. Conf. on Distributed Computing Systems,* Huntsville, AL, pp. 15-21, Oct. 1979.

[King81] King, J. J., "QUIST: A system for semantic query optimaization in relational databases," *Proc. Int. Conf. on Very Large Data Bases,* pp. 510-517, 1981.

[KiPo81] Kilov, K. I. and Popova, I. A., "Meta-database architecture for relational DBMS," *SIGMOD Record,* Vol. 12, No. 1, pp. 18-25, 1981.

[KoHa69] Kowalski, R. and Hayes, P., "Semantic trees in automatic theorem proving," *Machine Intelligence* (B. Meltzer and D. Michie, eds.), Vol. 4, American Elsevier, New York, pp. 87-101, 1969.

[KoKu70] Kowalski, R. and Kuehner, P., "Linear resolution with selection function," Metamathematics Unit, Edinburgh University, Scotland, 1970.

[Kowa74] Kowalski, R., "Predicate logic as a programming language," *Proc. IFIP Congress* (Information Processing 74), North-Holland, Amsterdam, pp. 569-574, 1974.

[KrKr67] Kreisel G. and Krivine J. L., *Elements of Mathematical Logic (Model Theory),* North-Holland, Amsterdam, 1967.

[Lee 72] Lee, R. C. T., "Fuzzy logic and the resolution principle," *J. ACM,* Vol. 19, No. 1, pp. 109-119, Jan. 1972.

[Love70] Loveland, D. W., "A linear format for resolution," *Proc. IRIA Symp. on Automatic Demonstration,* Versailles, France, 1968, Springer-Verlag, New York, pp. 147-162, 1970.

[Love72] Loveland, D. W., "A unifying view of some linear Herbrand procedures," *J. ACM,* Vol. 19, pp. 366-384, March 1972.

[Luck70] Luckham, D., "Refinements in resolution theory," *Proc. IRIA Symp. on Automatic Demonstration,* Versailles, France, 1968, Springer-Verlag, New York, pp. 163-190, 1970.

[MaRi76] Mahmoud, S. and Riordon, J. S., "Optimal allocation of resources in distributed information networks," *ACM Transactions on Database Systems,* Vol. 1, No. 1, pp. 66-78, Mar. 1976.

[MaUl83] Maier, D. and Ullman, J. D., "Fragments of relations," *Proc. ACM SIGMOD Int. Conf. on Management of Data,* San Jose, CA, pp. 15-22, May 1983.

[McDe82] McDermott, D., "A temporal logic for reasoning about processes and plans," *Cognitive Science,* Vol. 6, 1982.

[McHa69] McCarthy, J. and Hayes, P. J., "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence* (D. Michie and D. Meltzer, eds), Vol. 4, Edinburgh University Press, Edinburgh, Scotland, pp. 463-502, 1969.

[McMi77] McSkimin, J. R. and Minker, J., "The use of a semantic network in a deductive question anwering system," *Proc. Int. Joint Conf. on Artificial Intelligence,* Cambridge, MA, 1977.

[Melt66] Meltzer, B., "Theorem-proving for computers: Some results on resolution and renaming," *Computer J.,* Vol. 8, pp. 341-343, 1966.

[Mink78] Minker, J., "Experimental relational data base system bases on logic," *Logic and Databases* (H. Gallaire and J. Minker, eds), Plenum Press, New York, pp. 107-148, 1978.

[Mins75] Minsky, M., "A framework for representing knowledge," *The Psychology of Computer Vision* (P. Winston, ed), McGraw-Hill, New York, pp. 211-277, 1975.

[MoLe77] Morgan, H. L. and Levin, K. D., "Optimal program and data locations in computer networks," *Comm. ACM,* Vol. 32, No. 5, pp. 315-322, May 1977.

[Mylo81] Mylopoulos, J., "An overview of knowledge representation," *Proc. Workshop on Data Abstraction, Databases and Conceptual Modeling,* pp. 5-12, Jan. 1981.

[NeSi72] Newell, A. and Simon, H. A., *Human Problem Solving,* Prentice-Hall, Englewood Cliffs, NJ, 1972.

[Nils80] Nilsson, N. J., *Principles of Artificial Intelligence,* Tioga Pub. Co., 1980.

[NoRu75] Norman, D. A., Rumelhart, D. E. and the LNR Research group, *Explorations in cognition,* Freeman Pub. Co., San Francisco, CA, 1975.

[Ouli84] Oulid-Aissa, M., *The Distribution and Materialization of Cross-Referencing Data Units in a Computer Network,* Ph. D. Dissertation, The Univ. of Michigan, Ann Arbor, MI, 1984.

[Quil68] Quillian, M. R., "Semantic memory," *Semantic Information Processing* (M. Minsky, ed), MIT Press, Cambridge, MA, pp. 227-270, 1968.

[Quin55] Quine, W. V., "A Proof Procedure for Quantification Theory," *J. Symbolic Logic,* Vol. 20, pp. 141-149, 1955.

[Raph68] Raphael, B., "SIR: A computer program for semantic information retrieval," *Semantic Information Processing* (M. Minsky, ed), pp. 33-145, 1968.

[RaWa79] Ramamoorthy, C. V. and Wah, B. W., "The placement of relations in a distributed relational database," *Proc. Int. Conf. on Distributed Computing Systems,* Huntsville, AL, Oct. 1979.

[Rebo76] Reboh, R. et. al., "QLISP: A language for the interactive development of complex systems," Rep. No. TN-120, AI Center, SRI Int., Inc., 1976.

[Reit71] Reiter, R., "Two results on ordering for resolution with merging and linear format," *J. ACM,* Vol. 18, pp. 630-646, 1971.

[Reit78a] Reiter, R., "On Reasoning by Default," *Proc. TINLAP-2.,* The University of Illinoi, Urbana, IL, July 1978.

[Reit78b] Reiter, R., "Deductive Question-Answering on relational data bases," *Logic and Databases* (H. Gallaire and J. Minker, eds), Plenum Press, New York, pp. 149-177, 1978.

[Reit81] Reiter, R., "On the integrity of typed first order data bases," *Advances in Data Base Theory* (H. Gallaire, J. Minker and J. M. Nicolas, eds), Plenum Press, New York, 1981.

[Robi65a] Robinson, J. A., "A machine-oriented logic based on the resolution principle," *J. ACM*, Vol. 12, No. 1, pp. 23-41, Jan. 1965.

[Robi65b] Robinson, J. A., "Automatic deduction with hyper-resolution," *Int. J. Comput. Math.*, Vol. 1, pp. 227-234, 1965.

[RoGo77] Rothnie, J. B. and Goodman, N., "A survey of research and development in distributed database management," *Proc. Int. Conf. on Very Large Data Bases*, Tokyo, Japan, pp. 48-62, Oct. 1977.

[Roth80] Rothnie, J. B. et. al., "Introduction to a system for distributed databases (SDD-1)," *ACM Transactions on Database Systems*, Vol. 5, No. 1, pp. 1-17, March 1980.

[RuDW72] Rulifson, J., Derkson, J. A. and Waldinger, R. J., "QA4: A procedural calculus for intuitive reasoning," Rep. No. TN-83, AI Center, SRI Int., Inc., 1972.

[SAMP81] *Proc. ACM SIGART/SIGMOD/SIGPLAN workshop on Data Abstraction, Databases and Conceptual Modelling,* Pingree Park, Col., 1981.

[Schm38] Schmidt, A., "Uber deduktiven Theorien mit mehreren Sorten von Grunddingen," *Mathematische Annalen*, Vol. 115, pp. 485-506, 1938.

[Schm51] Schmidt, A., "Die Zulassigkeit der Behandlung mehrsortiger Theorien mittels der ublichen Pradikatenlogik," *Mathematische Annalen*, Vol. 123, pp. 187-200, 1951.

[Schu76] Schubert, L. K., "Extending the expressive power of semantic networks," *Artificial Intelligence*, Vol. 11, No. 1, 2, pp. 45-83, 1976.

[Shap79] Shapiro, S., "The SNePS semantic network processing system," *Associative Networks -- The Representation and Use of Knowledge in Computers*, Academic Press, New York, pp. 179-203, 1979.

[ShIr84] Shin, D. G. and Irani, K. B., "Knowledge representation using an extension of a many-sorted language," *Proc. Conf. on Artificial Intelligence Applications*, Denver, Col., pp. 404-409, Dec. 1984.

[Shoe67] Shoenfield, J. R., *Mathematical Logic*, Addison-Wesley, Reading, MA, 1967.

[Shor76] Shortliffe, E. H., *Computer-Base Medical Consultations: MYCIN*, North-Holland, New York, 1976.

[Siek84] Siekman, J., "Universal unification", *Proc. Int. Conf. on Automated Deduction,* Napa, CA, (Lecture Notes in Computer Science, Vol. 170), Springer-Verlag, New York, pp. 1-42, 1984.

[Slag67] Slagle, J. R., "Automatic theorem proving with renameable and semantic resolution," *J. ACM,* Vol. 14, pp. 687-697, March 1967.

[SmSm77] Smith J. M. and Smith D. C. P., "Database abstractions: aggregation and generalization," *ACM Transactions on Database Systems,* Vol. 2, No. 2, pp. 105-123, 1977.

[SmSm78] Smith J. M. and Smith D. C. P., "Principle of conceptual DB design," *Proc. NYU Symp. on DB Design,* New York, pp. 18-19, May 1978.

[SuMc72] Sussman, G. and McDermott, D. V., "CONNIVER reference manual," Memo 259, AI Laboratory, Massachusetts Institute of Technology, 1972.

[TeFr82] Teorey, T. J. and Fry, J. P., *Design of Database Structures,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[Ullm80] Ullman, J. D. *Principles of Database Systems,* Computer Science Press, Rockville, Md., 1980.

[Walt78] Waltz, D. L., "An English language question answering system for a large relational database," *Comm. ACM,* Vol. 21, No. 7, pp. 526-539, 1978.

[Walt83] Walther, C., "A many-sorted calculus based on resolution and paramodulation," *Proc. Int. Joint Conf. on Artificial Intelligence,* Karlsruhe, West Germany, 1983.

[Walt84a] Walther, C., "Unification in many-sorted theories," *Proc. European Conf. on Artificial Intelligence,* Pisa, Italy, 1984.

[Walt84b] Walther, C., "A mechanical solution of Schubert's streamroller by many-sorted resolution," *Proc. National Conf. on Artificial Intelligence,* Austin, TX, pp. 330-334, 1984.

[Wang52] Wang, H., "Logic of many-sorted theories," *J. Symbolic Logic,* Vol. 17, No. 2, pp. 105-116, June 1952.

[Wang60] Wang, H., "Towards mechanical mathematics," *IBM J. Research Dev.,* Vol. 4, pp. 2-22, 1960.

[Weyh77] Weyhrauch, R. W. "FOL, a proof checker for first-order logic," Memo AIM-235.1, Stanford Artificial Intelligence Laboratory, Stanford University, 1977.

[Whit70] Whitney, V. K. M., "A study of optimal file assignment and communication network configuration in remote-access computer message processing and communication systems," SEL Tech. Report No. 48, The Univ. of Michigan, Ann Arbor, MI, Sept. 1970.

[Wins77] Winston, P. H., *Artificial Intelligence,* Addison-Wesley, Reading, MA, 1977.

[WoCR64] Wos, L., Carson, D. F. and Robinson, A., "The unit preference strategy in theorem proving," *Proc. AFIPS Fall Joint Computer Conf.,* Vol. 26, pp. 616-621, 1964.

[WoKa83] Wong, E. and Katz, R. H., "Distributing a database for parallelism," *Proc. ACM SIGMOD Int. Conf. on Mangement of Data,* San Jose, CA, pp. 23-29, May 1983.

[WoMy77] Wong, H. K. T. and Mylopoulos, J., "Two views of data semantics: A survey of data models in artificial intelligence and database mangement," *Information,* Vol. 15., No. 3, pp. 344-383, 1977.

[Wong77] Wong, E., "Retrieving dispersed data from SDD-1: A system for distributed databases," *Proc. Berkeley Workshop on Distributed Data Management and Computer Networks,* The Univ. of California, Berkeley, CA, pp. 217-235, May 1977.

[Wong81] Wong, E., "Dynamic re-materialization: Processing distributed queries using redundant data," *Proc. of Berkeley Workshop on Distributed Data Management and Computer Networks,* The Univ. of California, Berkeley, CA, pp. 3-13, 1981.

[WoRC65] Wos, L., Robinson, A. and Carson, D. F., "Efficiency and completeness of the set of support strategy in theorem proving," *J, ACM,* Vol. 12, pp. 536-541, March 1965.

[WoYo76] Wong, E. and Youssefi, K., "Decomposition -- A strategy for query processing," *ACM Transactions on Database Systems,* Vol. 1, No. 3, pp. 223-241, Sept. 1976.

[Yao 79] Yao, S. B., "Optimization of query evaluation algorithms," *ACM Transactions on Database Systems,* Vol. 4, No. 2, pp. 133-155, June 1979.