HL--RESOLUTION:

A SEMANTIC REFINEMENT OF RESOLUTION
AND A THEORY OF MODEL SPECIFICATION

BY: David M. Sandford
DCS-TR-81

#160

Dept. of Computer Science
Rutgers, The State Univ.
New Brunswick, NJ

This work is supported in part by the
Advanced Research Projects Agency of
the Office of the Secretary of Defense
under grant DAHCIS-73-C-06.

July 1979
© 1979

David Michael Sandford

ALL RIGHTS RESERVED
HL--RESOLUTION:
A SEMANTIC REFINEMENT OF RESOLUTION
AND A THEORY OF MODEL SPECIFICATION

by DAVID MICHAEL SANDFORD

A thesis submitted to
The Graduate School
of
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Computer Science

Written under the direction of
Professor Chitoor V. Srinivasan

and approved by
C. V. Srinivasan
Ann Yasuhara
Don Loveland
Robert Smith

New Brunswick, New Jersey
October 1979
ABSTRACT OF THE THESIS

HL- Resolution:
A Semantic Refinement of Resolution
and a Theory of Model Specification
by DAVID MICHAEL SANFORD, Ph.D.

Thesis Director: Professor Chitoor V. Srinivasan

Two related but independent topics are treated dealing with methods for mechanical theorem proving in the first order predicate calculus using the resolution system of J. A. Robinson. Current research efforts are almost totally oriented towards reducing the amount of effort required by existing methods to prove theorems of practical interest, and this thesis is a theoretical treatment of material relating to efficiency.

The first topic of the thesis deals with a new strategy, called Hereditary Lock Resolution (HLR) which is a refinement of the original resolution inference rule. HLR is composed of two interacting refinements. One is a modification of the locking refinement (a syntactic refinement) of Boyer, and the other is a strengthening of the model strategy (a semantic refinement) of Luckham. Previously known strategies combining syntactic and semantic components either used a weaker syntactic strategy than lock resolution, or used weaker semantic notions, or were incomplete (i.e. unable to prove some theorems). HLR is complete and sound (i.e. never constructs fallacious
proofs). HLR generates a search space involving clauses, as does ordinary resolution, but each clause has attached to it an additional data structure which contains information about the derivation leading to that clause. This data structure is called an FSL (False Substitution List) and consists of a set of literals all of which must be falsifiable according to some model (which initially can be chosen arbitrarily). The FSL mechanism is applicable to other semantic refinements of resolution besides HLR, and this is illustrated specifically for the case of the model strategy of Luckham.

The second topic of the thesis concerns the specification and use of models in resolution inference systems. The usual requirement in semantic refinements of resolution has been that the model used must be a Herbrand interpretation, which is an abstractly defined way of considering models. However, in pragmatic situations where implemented procedures must utilize models, Herbrand interpretations which capture the relevant structure of the domain to be modeled usually are both difficult to find and computationally costly to use. The thesis takes the position that the essence of the difficulty is that Herbrand interpretations require the specification of details which are mostly irrelevant to the theorem proving task, and that the way out of this difficulty is to develop a theory of models which are based on incomplete specification. The key to doing this is to focus on the interface between a
semantic refinement of resolution (e.g. HLR) and the model. This interface is simple and is adequately summarized by the notion of a semantic function, which is a function mapping logical sentences into the values "true" or "false". Once this is done a simple theory of incompletely specified models can be developed which defines semantic functions with the appropriate properties. The theory is both simple enough and detailed enough so that completeness of HLR and other semantic refinements can be demonstrated using these semantic functions instead of the usual Herbrand interpretations.

The efficiency problem in first order mechanical theorem proving is a multifaceted and difficult issue, and the results and views presented in this thesis are to be considered not as final solutions, but instead as contributions to the perspective and knowledge base from which more adequate strategies will ultimately be formulated.
ACKNOWLEDGMENTS

Foremost I wish to thank my adviser Professor Chitoor V. Srinivasan for patience, faith, and encouragement in a (seemingly) endless writing process.

My sincerest appreciation to Professor Anu Yasuhara for careful readings of the material of this thesis, both in its present and in previous forms, and to Professor Robert Smith for serving on the thesis committee under difficult circumstances.

Special thanks are due to Professor Donald W. Loveland of Duke University for literally (physically) going out of his way to serve on the thesis committee.

I also thank Professors Martin Davis of the University of California and Wolfgang Bibel of the Technische Universität München for reading and offering comments on this thesis.

This work was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under grant DAHCIS-73-C-66.
CONTENTS

Abstract ......................................................... ii
Acknowledgments ................................................. v
List of Figures .................................................. viii

Chapter 1

1.0 Introduction .................................................. 1
1.1 Overview ...................................................... 4
1.2 Conventions and Definitions ............................... 8

Chapter 2

2.0 Hereditary Lock Resolution ............................... 12
2.1 Examples of HL-Resolution ................................. 13
2.2 Combining LR and TMS: The Combination Issue .... 34
2.3 TMS and History Tracing ................................... 43
2.4 Formal Definition of HL-Resolution ..................... 53
2.5 Soundness of HL-Resolution ............................... 77

Chapter 3

3.0 Completeness of HL-Resolution ............................ 81
3.1 Completeness Theorems ...................................... 81
3.2 Herbrand Model Completeness Proof ..................... 84
3.3 Completeness of the Inference Rule HLR – t.v.d. .... 103
3.4 HLR False Permissive Completeness Proof ............. 110
Chapter 4

4.0 Models ......................................................... 114
4.1 Notation and Preliminary Definitions ....................... 118
4.2 Theoretical Construction
    of Sound Semantic Functions .......................... 127
4.3 Example III -- A Simple Model Structure .................. 143
4.4 Pragmatic Construction
    of Sound Semantic Functions .......................... 149
4.5 Example IV -- A Sophisticated Model ....................... 164
4.6 General Discussion of Models .............................. 193

Chapter 5

5.0 Discussion of HLR ........................................... 207
5.1 Refinement Strength of HLR ................................ 207
5.2 Overhead of HLR ........................................... 215
5.3 Experimental Results -- Domain of Applicability .......... 219
5.4 Compatibility with other Strategies ...................... 222
5.5 Areas for Future Research ................................. 227

Bibliography .................................................... 229
Table of Symbol Conventions .................................. 234
Index of Definitions ......................................... 237
Vita .............................................................. 240
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1--1</td>
<td>A TMS Search</td>
<td>5</td>
</tr>
<tr>
<td>2--1</td>
<td>Example I</td>
<td>19</td>
</tr>
<tr>
<td>2--2</td>
<td>Example II</td>
<td>25</td>
</tr>
<tr>
<td>2--3</td>
<td>A TMS + FSL Search</td>
<td>45</td>
</tr>
<tr>
<td>2--4</td>
<td>An Infeasible Deduction</td>
<td>45</td>
</tr>
<tr>
<td>4--1</td>
<td>Relationship of Herbrand Interpretations and Model Schemes in Different Languages</td>
<td>126</td>
</tr>
</tbody>
</table>
"There are no solved problems; there are only problems that are more or less solved."

H. Poincare
Chapter 1

INTRODUCTION

Despite limited success to date there remains strong interest in mechanizing theorem proving for the first order predicate calculus. The areas of application of a pragmatically effective automatic theorem prover include the purely logical (Morgan, 1976), as well as areas of considerable economic importance, such as program verification, and areas of application in artificial intelligence such as reasoning about actions. While in certain proposed application areas there remain unresolved issues concerning the appropriateness of a first order theorem proving approach (and sometimes of a theorem proving approach altogether) the primary difficulties are related to the efficiency of current techniques, even in those areas where the first order logic seems quite appropriate. The efficiency problem is currently a totally debilitating problem with respect to achieving most pragmatic applications.

As humans would intuitively judge problem complexity, the theorems arising in many application areas seem only several times harder than what can currently be handled by automatic methods. Thus one would expect that relatively modest efficiency increases (or even increases in machine hardware capability) would bring application areas into the
realm of both economic feasibility and real-time feasibility. This is not the case, however. Human intuitive estimates of how difficult it is to prove a particular theorem are based on features quite distinct from the features of theorems which determine the difficulty of proof finding for current automatic techniques. The level of theorem difficulty that can be handled by current techniques is that level that can be just proven before driving the search space into a combinatoric explosion of search possibilities. In such a situation, even small increases in theorem difficulty by human standards involve massive increases in effort to prove the theorem by automatic procedures. For this reason the only way that automatic theorem proving can become a pragmatic tool is through techniques which reduce the search space size, and not by increasing the rate at which a given search space is explored.

For several reasons the resolution inference rule of J. A. Robinson (Robinson, 1965) seems to be the most promising current approach to dealing with the efficiency issue. These reasons are not concerned with the intrinsic efficiency of the resolution inference system, but rather with its relative tractability in being analyzed and modified, and include:

1. the simplicity of the inference rule (consisting of only 1 or 2 primitive types of operations);
ii. the uniformity and simplicity of the logical statements in the search space (disjunctions of literals, with only universal quantification);
iii. the uniformity of the proof recognition criteria (generation of the empty clause).

There are three categories of strategy that have been investigated in the context of reducing the amount of work a resolution theorem prover does in searching for a proof of a theorem. The first is that of ordering strategies, which leave the underlying inference space unchanged, but expand that space in a manner (it is hoped) that will find the proof quickly. The second category is that of deletion strategies, which delete certain clauses in the search space by virtue of their being recognized (by some reasonably low overhead recognition procedure) as either irrelevant to the proof or redundant with respect to other clauses in the search space.

This thesis is concerned with a third category of strategies, called refinements. A refinement is a condition or property that deductions must satisfy for them to be part of the inference space. When we say that the resolution rule (or system) \( R_1 \) is a refinement of the resolution rule \( R_2 \), we mean that all of the deductions which are allowed by \( R_1 \) are also allowed by \( R_2 \). We call the original resolution rule (Robinson, 1965) unrestricted resolution. We will only consider inference rules, \( R \), which are refinements of
unrestricted resolution, and thus we simply call \( R \) a refinement, instead of explicitly saying \( R \) is a refinement of unrestricted resolution. There have been many refinements proposed ([Chang and Lee, 1973] covers many of these) and empirical evidence indicates that they play a crucial role in overall efficiency.

1.1 Overview.

The overall emphasis and layout of the thesis is as follows. The concentration in the thesis is on a particular refinement strategy, called Hereditary Lock Resolution (HLR or HL-resolution), and related matters concerning the specification and use of models in the theorem proving task.

HL-resolution is a refinement combining a syntactic and a semantic component. The syntactic component is a modification of the Lock Resolution (LR) refinement (Boyer, 1971), which is one of the strongest of the known, general purpose, syntactic refinements. The semantic component is an extension of The Model Strategy (TMS) of Luckham (Luckham, 1970). The extension transforms TMS from a local condition on parts of a deduction to a global condition on entire deductions.
\[ \mathcal{M} = \{ \neg P(a), P(b), \neg Q, R, S \} \]

1. \( \neg P(x), Q(x) \); \hspace{1cm} \text{false} \\
2. \( \neg Q(x), R(x) \); \hspace{1cm} \text{true} \\
3. \( \neg R(a) \); \hspace{1cm} \text{false} \\
4. \( P(x), S(x,a,b) \); \hspace{1cm} \text{true} \\

\text{level 1} \\
2tx1f = 5. \( \neg P(x), R(x) \); \hspace{1cm} \text{true} \\
3fx2t = 6. \( \neg Q(a) \); \hspace{1cm} \text{true} \\
4tx1f = 7. \( Q(x), S(x,a,b) \); \hspace{1cm} \text{true} \\

\text{level 2} \\
5tx3f = 8. \( \neg P(a) \); \hspace{1cm} \text{true} \\
6tx1f = 9. \text{ duplicate of 8.} \\

**Figure 1** -- 1 A TMS Search
In Figure 1--1 we have a small satisfiable set of clauses along with a complete listing of the search space using TMS as the refinement with the indicated model, \( M \) (where for those literals all of whose ground instances are true we just write the literal letter). At level 2 it is seen that there are two distinct deductions of the clause "\(-P(a)\);" (we use \(-\) as the negation sign). This is an obvious redundancy in search effort, and in more complicated clause sets such a redundancy can have a large effect on efficiency. Example 1 of Chapter 2 takes the clause set and model shown in Figure 1--1, and expands the search space according to HLR. There it is seen that the redundancy does not occur in HLR for this clause set. Chapter 2 presents the HLR refinement in detail, first through examples and an informal treatment, after which it is defined in a rigorous way, and finally the soundness of HLR is demonstrated. In addition some of the components of HLR are discussed in isolation in order to make their characteristics more intuitively clear.

Chapter 3 is concerned with demonstrating the completeness of HLR. Completeness is shown to hold not only for the use of a Herbrand interpretation as the model, but also for other types of models.

Chapter 4 presents a theory of model specification and use in resolution semantic refinements which allows considerable flexibility and sophistication of models in
pragmatic environments. This chapter is relatively independent of the other chapters.

Chapter 5 is a general discussion and summary giving a perspective on the HLR refinement, and indicates what seem to be important areas for future investigation.

This thesis is not intended to serve in any fashion as a primer in resolution theory, and is thus not accessible to the reader with no background in resolution. The required background material is almost entirely available between (Chang and Lee, 1973) and (Nilsson, 1971), with the assumption that the reader already has some familiarity with the first order predicate calculus. Background material in resolution not covered in the two sources above will be given specific references when the material is first used in the text, but in most cases the main idea of the text should be clear even without going to the references.
1.2 Conventions and Definitions.

In this section we define several items which will be needed later.

We use the term input set to designate the initial set of clauses under consideration.

A Herbrand interpretation, \( h \), for the language \( L \), is defined to be the (usually infinite) set of ground literals
\[
\{ m_1, m_2, \ldots \}
\]
where each \( m_i \) is either \( A_i \) or \( \neg A_i \), and

\( A_1, A_2, \ldots \)
is any non-redundant enumeration of the atoms in the Herbrand base of \( L \). Herbrand base and Herbrand universe are defined as usual (see specifically, Chang and Lee, 1973), but notice that we will refer to them as the Herbrand base (universe) "for the language", instead of the more usual "for the clause set". Also notice that the Herbrand base is to contain all possible ground atoms in the language, and not just those that are ground instances of atoms appearing in the clause set.

A resolution inference system is said to be refutation complete iff for every set of unsatisfiable clauses, there exists a refutation in the inference system. When we just use the word complete, refutation complete is to be understood. Refutation completeness is an important property since in this thesis we only consider the use of
resolution as a refutation procedure.

In contrast to refutation completeness is the notion of deduction incompleteness. A resolution refinement $R_1$ is said to be deduction incomplete with respect to the resolution system $R_2$ iff

i. $R_1$ is a refinement of $R_2$;

ii. there exists a clause set $S$, and a clause $C$ derivable from $S$ in $R_2$ such that $C$ is not subsumed by any clause derivable from $S$ by $R_1$.

We speak loosely of the "amount" of deduction incompleteness of $R_1$ (relative to $R_2$), and mean by this the size of the set of clauses deducible by $R_2$ but not subsumed by any clause deducible by $R_1$.

A set of clauses, $S$, is said to be minimally unsatisfiable iff both

i. $S$ is unsatisfiable;

ii. every proper subset of $S$ is satisfiable.

We will use the term TMS (The Model Strategy) to denote the refinement which requires each resolvent to have at least one false parent. The truth values are determined according to an arbitrarily chosen (but fixed for the duration of the search) Herbrand interpretation. The Herbrand interpretation is to be a Herbrand interpretation for the language of the input set, and a clause is false iff it has a ground instance (over the Herbrand universe) such
that none of its ground literals is in the interpretation.

The above description of TMS is essentially the description given in (Nilsson, 1971), where it is attributed to (Luckham, 1970). However (Luckham, 1970) uses a different definition of interpretation than used here. When applied to ground input sets, TMS and Luckham's model strategy are identical, since Luckham's model then becomes a Herbrand interpretation for the language of the input set (save for a distinction we can ignore for the purposes of this discussion). Thus ground TMS is contained in Luckham's model strategy. It is a simple matter to lift ground TMS to the general level (and is done in Theorem 2.2-3 of this thesis), and thus the completeness of TMS as defined above is a simple and immediate consequence of (Luckham, 1970). It is for this reason that we concur with (Nilsson, 1971) in attributing TMS to (Luckham, 1970).

The completeness of TMS is also an immediate consequence of the completeness of Slagle's semantic P-deduction (Slagle, 1967). Slagle's interpretation for a set of clauses, S, which he calls a latent model, is a subset of what we call a Herbrand interpretation for S. For the present purposes this difference in interpretations is of no concern. Semantic P-deduction involves a predicate letter ordering of literals in false clauses, and is defined in a clash framework. It is easy to show that semantic P-deduction is a refinement of TMS, and thus the
completeness of semantic P-deduction gives the completeness of TMS.

We will use the phrase semantic resolution to refer mainly to semantic P-deduction, but also to other clash oriented rules of Slagle (Slagle, 1967, 1972).

Deduction trees are to be thought of as rooted at the bottom and with leaves at the top. We will be concerned with resolution inference rules which combine exactly two clauses at a time (called parents) to obtain a new clause (called the resolvent). Thus the deduction trees of interest in this thesis are strict binary trees, i.e. trees where each internal node has exactly two parent nodes. A deduction tree of exactly three (connected) nodes is also called an inference step.

Whenever it is appropriate (e.g. in resolving two clauses) it will be assumed, without explicit mention, that the expressions involved have already had their variable symbols standardized apart.

Because of constraints on the available character set some simulation of common symbols has been done. There is a table just before the index of this thesis which identifies these symbols along with other notational conventions.
Chapter 2

HEREDITARY LOCK RESOLUTION

This chapter presents a complete resolution refinement, called Hereditary Lock Resolution (HLR, or HL-resolution), which combines lock resolution (Boyer, 1971) with the model strategy (Luckham, 1970). HLR obtains strength as a refinement through both the syntactic ordering of literals provided by lock resolution, and by the focusing effect (on the false clauses) provided by the model strategy.

In order to obtain an intuitive understanding of HLR the reader should bear in mind (particularly for the examples to follow) that the refinement deals simultaneously with three essentially independent issues. These issues are:

i. The Combination Issue: The accommodation of lock resolution (LR) and the model strategy (TMS), into a refutation complete refinement.

ii. The History Tracing Issue: The use of a notational device (called the False Substitution List (FSL)) as a semi-global constraint mechanism.

iii. The Model Scheme Issue: The use of a generalized notion of a model, which, while intuitively natural, entails some additional complexity when demonstrating the completeness of HLR.
The exposition of HLR will take the following course. First some examples of HLR will be given which involve all three of the above issues. These examples are intended to illustrate what HLR looks like in an implementation sense as a refinement of resolution. The only distinction between the examples and intended implementation versions of HLR are that the examples totally ignore search ordering strategies (breadth first search is used), almost totally ignore deletion strategies, and do no factoring.

Secondly attention will be turned to the combination issue for LR and TMS at the ground level. Thirdly the history tracing issue will be dealt with by giving an example in the context of TMS at the general level. These treatments of the combination issue and history tracing issue should make sense to the reader in the context of the HLR examples seen previously.

Finally in this chapter a precise definition of the HLR refinement is given, and the soundness properties are stated and proved for HLR.

2.1 Examples of HL-Resolution.

As a procedure, HLR accepts a set of normal clauses as the input set, along with a model for the language of the clauses, and transforms the input set into HL-clauses. The inference rule of HLR is then applied to generate new
clauses, and the generation of the null clause signals the unsatisfiability of the original input set.

After some initial general remarks about this procedure, the procedure will be further explained in the context of two examples.

2.1.1 Transforming the Input Set

The input set has two integers assigned to each literal. One of these is to be called the true lock number, and the other the false lock number of the literal.

If \( \mathcal{C} \) is a set of clauses with double lock numbers assigned to its literals, then we say that \( \mathcal{C} \) has an HL-proper lock numbering iff for all literals \( \bar{L}, \bar{L}' \) in \( \mathcal{C} \), the true lock number of \( \bar{L} \) is less than the false lock number of \( \bar{L}' \). In HLR we are only interested in clause sets with HL-proper lock numberings. There are no other constraints governing how lock numbers are assigned to literals in HLR; however, as a matter of convention, uniformity, and desire to avoid unconscious bias, all examples of input clause sets have been lock numbered by a simple and obvious algorithm (e.g. obvious from the input sets listed in Figures 2-1 and 2-2).

After assigning lock numbers to literals each clause has associated with it the empty list. This list is called the False Substitution List (FSL) of the clause. The FSL of
a clause is the empty list iff the clause is an input clause.

Finally each clause has assigned to it a truth value designator (t.v.d.) which is one of "T", "F", or "T/F". These t.v.d. values are determined by the model which is supplied along with the input set. Their computation will be illustrated later.

2.1.2 The HLR Inference Rule.

HLR produces resolvents by combining two parent clauses at a time. In the examples to follow no factoring will be done, so that this will be a binary resolution inference rule (Chang and Lee, 1973).

In each inference step one of the parents is identified as a false parent and the other as a true parent. The inference rule is asymmetrical in the two parents, and we say the false parent is being resolved against the true parent. A requirement in HLR is that the true parent have T or T/F as its t.v.d., while the false parent must have a t.v.d. value of F or T/F.

Each resolvent produced has double lock numbers on its literals which are inherited from the parent literals. In addition each resolvent has an FSL constructed for it which contains certain literals from the parent clauses.
An FSL is said to be feasible (and the clause it belongs to is said to be feasible) iff there exists a substitution which simultaneously converts every literal in the FSL into a ground literal which is false in the model. The feasibility condition on clauses can be thought of as a requirement for a certain type of relevancy (to the proof) for each clause. This relevancy is tested by a mechanism that is both global and semi-global. Feasibility is a global property in the sense that a single model is used for all of the feasibility tests. Feasibility is a semi-global property in the sense that the FSL literals tested in a clause are in the clause because of the deduction tree leading to that clause.

The HLJ refinement is essentially implemented by two mechanisms:

1. the use of the double lock numbers to limit the possible choices of which literals to resolve on;
2. the requirement that every clause kept in the search space must be feasible.

It will be seen that there is an interaction between lock numbers and feasibility testing in that the content of the FSL of a derived clause is partially dependent upon how lock numbers were assigned in the input set.

The above general description will be explained in more detail in the examples to follow.
2.1.3 Example I.

Figure 2--1 shows a set of 4 input clauses resulting from transforming the 4 input clauses of Figure 1--1 into a set of HL-clauses.

In order to obtain the t.v.d. values of clauses it is necessary to define several functions. The first two are the selector or projection functions $S$ and $F$, such that for a clause $H$,

$S(H)$ is the set of standard literals for $H$
$F(H)$ is the set of FSL literals for $H$.

Next we need a function, $\phi_h$, for the Herbrand interpretation $h$, which is the usual semantic function mapping sentences onto (true, false). A set of literals, such as the literals in a normal clause, or the FSL literals in an HL-clause, has a truth value under $\phi_h$ by considering the set to be the universal closure of the disjunction of its members. Thus, for $K$ a set of literals in the language $L$, and $\sigma$ ranging over substitutions for $L$,

$$\phi_h(K) = \begin{cases} 
\text{false} & \text{if } \exists \sigma : \{ (K\sigma \text{ is a set of ground literals}) \\
& \text{and } (K\sigma \cap h = \{\}) \} \\
\text{true} & \text{otherwise}
\end{cases}$$

where {} is the empty set. For the purposes of semantic evaluations any lock numbers on literals are invisible.
If \( \mathcal{H} \) is an NL-clause, and \( x_1, x_2, \ldots, x_m \) are all of the distinct variable symbols appearing in standard and FSL literals in \( \mathcal{H} \), then \( \forall^x \) is defined to be the quantifier list \( \forall x_1, \forall x_2, \ldots, \forall x_m \).

If \( K \) is a set of literals, then \( \lor(K) \) is the disjunction of the elements of \( K \).

Finally we need the definitions of two functions, \( \alpha \) and \( \beta \), which map NL-clauses to sentences:

\[
\begin{align*}
\alpha(\mathcal{H}) & \equiv \forall^x \{ (\neg \lor(\neg(\mathcal{H}))) \rightarrow \lor(\mathcal{U}(\mathcal{H})) \} \\
\beta(\mathcal{H}) & \equiv \forall^x \{ (\neg \lor(\neg(\mathcal{H}))) \rightarrow (\neg \lor(\mathcal{U}(\mathcal{H}))) \}.
\end{align*}
\]

\( \alpha(\mathcal{H}) \) and \( \beta(\mathcal{H}) \) are called the \( \alpha \)- and \( \beta \)-statements of \( \mathcal{H} \).

For a clause with an empty FSL, the expression \( \neg \lor(\neg(\mathcal{H})) \) is the empty conjunction, which we write as \( \Delta \).

If \( \mathcal{H} \) is a feasible NL-clause (i.e., if \( \phi_\mathcal{H}(\neg(\mathcal{H})) = \text{false} \)) then we define the truth value designator function

\[
t.v.d.(\mathcal{H}, \phi_\mathcal{H}) = \begin{cases} 
T & \text{if } \phi_\mathcal{H}(\neg(\mathcal{H})) = \text{true} \\
F & \text{if } \phi_\mathcal{H}(\neg(\mathcal{H})) = \text{true} \\
T/F & \text{otherwise}
\end{cases}
\]

We have not defined the truth value of an arbitrary sentence according to a Herbrand interpretation, \( h \), and thus we treat...
the application of $\phi_\lambda$ to the $\alpha$- and $\beta$-statements in an informal manner in these examples (the precise definition is given in section 2.4.8.6). Notice that it is not possible for both the $\alpha$- and $\beta$-statements to be true for the same clause when $\phi_\lambda$ is the usual semantic function for a Herbrand interpretation, $h$.

We can now illustrate the t.v.d. computations for the input clauses in Figure 2--1, as follows:

$\alpha$ (clause 1) = $\forall x \ (\neg A \rightarrow (\neg P(x) \lor Q(x)))$

$\beta$ (clause 1) = $\forall x \ (\neg A \rightarrow (P(x) \land \neg Q(x)))$

where lock numbers are not written on the literals, as they are not involved in the evaluation process. It is easily seen that both the $\alpha$- and $\beta$-statements are false in $M_\lambda$, so the t.v.d. of clause 1 is T/F.

$M_\lambda = \{\neg P(a), P(b), \neg Q, R, S\}$

1. $\neg P(x)_{\lambda_{001}}, Q(x)_{\lambda_{002}}$; FSL = {} T/F
2. $\neg Q(x)_{\lambda_{253}}, R(x)_{\lambda_{024}}$; FSL = {} T
3. $\neg R(0)_{\lambda_{005}}$; FSL = {} F
4. $P(x)_{\lambda_{006}}, S(x,a,b)_{\lambda_{007}}$; FSL = {} T

level 1

1Fx4T = 5. $Q(x)_{\lambda_{002}}, S(x,a,b)_{\lambda_{007}}$; FSL = $\{\neg P(x), Q(x)\}$ T

FIGURE 2--1 EXAMPLE 1
For clause 2 we have

\[ \alpha (\text{clause 2}) = \forall x \ (\bigvee \rightarrow (\neg Q(x) \lor \neg R(x))) \]
\[ \beta (\text{clause 2}) = \forall x \ (\bigwedge \rightarrow (Q(x) \land \neg R(x))) \]

and

\[ \phi_\alpha (\alpha (\text{clause 2})) = \text{true} \]
\[ \phi_\beta (\beta (\text{clause 2})) = \text{false} \]

giving \( t \cdot v \cdot d (\text{clause 2}, \phi_\alpha) = T \).

The \( t \cdot v \cdot d \) evaluations of clauses 3 and 4 are similarly easily obtained. Notice that the 4 input clauses are all trivially feasible since their FSL's are empty.

We now consider the breadth first search starting with these 4 clauses. We write \( iFxjT \) to mean that clause \( i \) is to be the false parent, and clause \( j \) the true parent, in an inference step.

Consider first \( 1Fx2T \). A rule in MLR is that the false parent can resolve only on a standard literal whose false lock number is not greater than the false lock number on any other standard literal in the clause. Thus in clause 1 the literal to be resolved on, called the selected literal, is \( \neg P(x) \)''. But this has no (unifiably) complementary matching literal in clause 2, so that \( 1Fx2T \) yields no resolvents.
The next possibility to consider is 3Fx1T, but again there are no complementary literals. Next is 3Fx2T, which does have a complementary literal match. We form the resolvent, Q1, of 3Fx2T as

\[ Q1 = \neg Q(a_{\neq 3}) \]

\[ FSL = \{ \neg R(a), \neg Q(a) \} \]

The rules for doing this are:

a. the standard literals of the resolvent, in this case "\( \neg Q(a_{\neq 3}) \)" are the same as would be obtained from the ordinary binary resolution step on the standard literals of the parents (except that lock numbers are inherited);

b. the FSL literals of the resolvent are the set union of all of the following:

i. the FSL's of each parent clause (empty in this case);

ii. all of the standard literals of the false parent, with lock numbers being dropped ("\( \neg R(a) \)" in this case);

iii. all of the standard literals from the true parent that have true lock numbers smaller than the true lock number of the literal being resolved on in the true parent, again with lock numbers being dropped (in this case "\( \neg Q(a) \)"").

The unifier used on the standard literals is also applied to the FSL literals (in this case the unifier is \( a/x \)).
We now check the feasibility of Q! and find that it is infeasible, and thus can be deleted from the search space. There are clearly no other literal matches for 3Fx2T, so that no resolvents at all can be generated from this pair of clauses. Notice that 3Tx2F is not allowed in HLRIA because the t.v.d. of clause 5 is F, and thus it cannot be the true parent (3Tx2F is also not allowed because clause 2 has a t.v.d. value of T, and thus cannot be the false parent).

The next possibility is 3Fx4T, which results in clause 5 of Figure 2--1. Clause 5 is feasible since
\[ \phi_k (\{ \neg P(x), Q(x) \}) = \text{false} \, . \]
The \( \alpha \)- and \( \beta \)-statements of clause 5 are:
\[ \alpha(\text{clause 5}) = \forall x [(P(x) \land \neg Q(x)) \rightarrow (Q(x) \lor S(x,a,b))] \]
\[ \beta(\text{clause 5}) = \forall x [(P(x) \land \neg Q(x)) \rightarrow (\neg Q(x) \land \neg S(x,a,b))] \]
and we see that
\[ \phi_k (\alpha(\text{clause 5})) = \text{true} \]
\[ \phi_k (\beta(\text{clause 5})) = \text{false} \]
so that t.v.d. (clause 5, \( \phi_k \)) = T.

The next pair of parents to consider is 3Fx4T, but there are no literal matches here. This exhausts the potential parent pairs among the input clauses, and thus we have completed level 1 of the search space.

In order to produce clauses at level 2 we must use clause 5, as a true parent, and find some other clause which meets two requirements:
i. has a t.v.d. value of F or T/F, so that it can be a false parent;  
ii. has a complementary literal match for one of the standard literals of clause 5.

There is no clause already in the search space which meets these two conditions. Thus the search terminates with just clause 5 being generated. The reader may wish to compare this HLR search space of Figure 2--1 with the TMS search space of Figure 1--1.

This concludes Example I.

2.1.4 Example II.

Our second example is the clause set used in example 6 of (Luckham, 1968), and also example 7 of (Chang, 1970). The theorem statement is:

"If a is a prime number, and \( a = b^2/c^2 \), then \( a \) divides \( b \)."

We take the same 7 input clauses as (Luckham, 1968), and these give rise to clauses 1 through 7 in Figure 2--2a, when transformed into HLR-clauses. The intended number theory interpretation for the symbols is as follows:

\[
\begin{align*}
P(x) & \implies x \text{ is a prime number} \\
M(x,y,z) & \implies x \times y = z \text{ (usual multiplication)} \\
D(x,y) & \implies x \text{ divides } y \\
S(x) & \implies x^2
\end{align*}
\]
and \( a, b, c \) are any natural numbers such that \( a = b^2/c^2 \), and the domain is the positive integers.

The input set is unsatisfiable and ELM as defined for these examples admits a refutation. The entire breadth first search space consists of 25 clauses (7 input clauses and 18 generated clauses which are retained).

The model used for this search is equivalent to a Herbrand interpretation for the language of the clauses, but will be presented in a somewhat different style as follows.

Consider a countable infinite set of primitive elements \( \mathcal{P} = \{ p_1, p_2, \ldots \} \). The domain or universe of our interpretation will be the power set of \( \mathcal{P} \). Thus individuals will be subsets of \( \mathcal{P} \). In particular we take

\[
\begin{align*}
a & \Rightarrow \{ p_1 \} \\
b & \Rightarrow \{ p_1, p_2 \} \\
c & \Rightarrow \{ p_2 \}
\end{align*}
\]

We interpret the remaining predicate and function symbols of the language of the clauses in terms of the usual notions of set theory:

\[
\begin{align*}
P(x) & \Rightarrow x \text{ is a singleton set} \\
M(x, y, z) & \Rightarrow x \cup y = z \\
D(x, y) & \Rightarrow x \subseteq y \\
S(x) & \Rightarrow x
\end{align*}
\]

This interpretation will be the model, \( \mathcal{M} \). We now illustrate how some of the input clauses have their t.v.d. values computed.
1. \( P(a) \) \( \vdash \) FSL = \{ \} T
2. \( M(a, S(c), S(b)) \) \( \vdash \) FSL = \{ \} T
3. \( M(x, x, S(x)) \) \( \vdash \) FSL = \{ \} T
4. \( "M(x, y, z) \) \( \vdash \) FSL = \{ \} T
5. \( M(x, y, z) \) \( \vdash \) FSL = \{ \} T
6. \( "P(x) \) \( \vdash \) FSL = \{ \} T
7. \( a, b \) \( \vdash \) FSL = \{ \} T

**Level 1**

7Fx5T = 8. \( M(a, y, b) \) \( \vdash \) FSL = \{ \} T

7Fx6T = 9. \( P(a) \) \( \vdash \) FSL = \{ \} T

7Fx6T = 10. \( P(a) \) \( \vdash \) FSL = \{ \} T

**Level 2**

8Fx4T = 11. \( M(y, a, b) \) \( \vdash \) FSL = \{ \} T

7Fx9T = 12. \( M(b, z, u) \) \( \vdash \) FSL = \{ \} T

9Fx1T = 13. \( M(b, z, u) \) \( \vdash \) FSL = \{ \} T

10Fx1T = 14. \( M(y, b, u) \) \( \vdash \) FSL = \{ \} T

**Level 3**

11Fx4T = effective duplicate of clause 6. \( M(y, a, b) \) \( \vdash \) FSL = \{ \} T

12Fx1T = 15. \( M(b, b, u) \) \( \vdash \) FSL = \{ \} T

13Fx3T = INFEASIBLE. \( D(a, S(b)) \) \( \vdash \) FSL = \{ \} T

13Fx4T = 16. \( M(b, b, u) \) \( \vdash \) FSL = \{ \} T

14Fx3T = INFEASIBLE. \( D(a, S(b)) \) \( \vdash \) FSL = \{ \} T

14Fx4T = 17. \( M(b, y, u) \) \( \vdash \) FSL = \{ \} T

**Figure 2 -- 2a Example II**
15Fx3T = 18. \( \neg D(a,S(b)) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b)) \} \) F

15Fx4T = 19. \( \neg M(b,b,u) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,u), \neg D(a,u) \} \) F

16Fx3T = INFEASIBLE. \( \neg D(a,S(b)) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b)), D(a,b) \} \)

16Fx4T = effective duplicate of 13.

17Fx3T = INFEASIBLE. \( \neg D(a,S(b)) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b)), D(a,b) \} \)

17Fx4T = 20. \( \neg M(y,b,u) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(y,b,u), \neg D(a,u), D(a,y), \neg M(b,y,u) \} \) F

18Fx5T = 21. \( \neg M(a,y,S(b)) \) level 5

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(a,y,S(b)), \neg D(a,S(b)), \neg M(a,y,S(b)) \} \) F

18Fx6T = 22. \( \neg P(a) \) level 6

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b), S(b)), \neg M(a,y,S(b)) \} \)

18Fx9T = 23. \( \neg P(a) \) level 9

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(y,b,u), \neg D(a,u), D(a,y) \} \) F

19Fx3T = exact duplicate of 18.

19Fx4T = exact duplicate of 19.

20Fx3T = INFEASIBLE. \( \neg D(a,S(b)) \) level 4

\( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b)), D(a,b) \} \)

20Fx4T = exact duplicate of 17.

21Fx2T = 25. □; \( FSL = \{ \neg D(a,b), \neg P(a), \neg M(b,b,S(b)), \neg D(a,S(b)), \neg M(a,S(c), S(b)) \} \) F
For clause 1 we have
\[ \alpha(\text{clause 1}) = (\mathcal{A} \longrightarrow P(a)) \]
\[ \beta(\text{clause 1}) = (\mathcal{A} \longrightarrow \neg P(a)) \]

In the language of the model these statements become
\[ \alpha_{\mathcal{M}}(\text{clause 1}) = (\mathcal{A} \longrightarrow \{(p_i) \text{ is a singleton set}\}) \]
\[ \beta_{\mathcal{M}}(\text{clause 1}) = (\mathcal{A} \longrightarrow \{(p_i) \text{ is not a singleton set}\}) \]

Obviously only the \( \alpha_{\mathcal{M}} \)-statement is true, so that the t.v.d. of clause 1 is \( T \). In complete analogy to \( \phi_{\mathcal{A}} \) being a semantic function mapping sentences to their truth values according to the Herbrand interpretation \( h \), we define \( \psi_{\mathcal{M}} \) to be a semantic function mapping sentences to their truth value according to \( \mathcal{M} \). The definitions of feasibility and t.v.d. value are unchanged except that \( \psi_{\mathcal{M}} \) replaces \( \phi_{\mathcal{A}} \) (N.B. this is correct in this specific example because of the structure of this model, but in more general cases the t.v.d. definition must be that of section 2.4.14). Thus we write t.v.d.(clause 1, \( \psi_{\mathcal{M}} \)) = \( T \).

For clause 3 we have
\[ \alpha_{\mathcal{M}}(\text{clause 3}) = \forall x (\mathcal{A} \longrightarrow (x \cup x = x)) \]
\[ \beta_{\mathcal{M}}(\text{clause 3}) = \forall x (\mathcal{A} \longrightarrow (x \cup x \neq x)) \]
giving t.v.d.(clause 3, \( \psi_{\mathcal{M}} \)) = \( T \).

Clause 6 is the most difficult of the 7 input clauses to evaluate, where we have:
\[ \alpha_{\mathcal{M}}(\text{clause 6}) = \forall x \forall y \forall z \forall u (\mathcal{A} \longrightarrow \)
\[ (x \text{ is not a singleton set}) \lor (y \cup z \not\in u) \]
\[ \lor (x \notin u) \lor (x \subseteq y) \lor (x \subseteq z) \]
\]
and

\[ \beta_m(\text{clause 6}) = \forall x \forall y \forall z \forall u \ [ W \rightarrow (x \text{ is a singleton set}) \land (y \cup z = u) \land (x \subseteq u) \land (x \not\subseteq y) \land (x \not\subseteq z)] . \]

The \( \mathcal{C}_m \)-statement is false, since, e.g., the valuation \( \langle p_1, p_2' \rangle \) for \( x \) makes the consequent of the implication false, while the antecedent is always true. The \( \alpha_m \)-statement is true, so that t.v.d. \( \psi_m(\text{clause 6}, \forall \mathcal{F}) \) = \( T \).

We are not concerned here with how difficult it might be to actually perform the model evaluations for the set theory model being used. This issue will be dealt with in Chapter 4.

We now turn our attention to the generated clauses in Figure 2--2. Consider the inference step generating clause 9. In this step clause 7 is resolved with the literal "\( \neg D(x,y)^{ns} \)" in clause 6. Clause 7 is the false parent. The FSL literals in clause 9 have two sources:

1. the standard literals from the false parent \( (\neg D(a,b), \text{in this case}); \)

2. the standard literals from the true parent whose true lock number is less than the true lock number of the literal resolved on \( (\neg E(a), \neg M(b,z,u), \neg D(a,u), \text{in this case, after unification}). \)

Clause 9 is shown to be feasible by showing that

\[ \forall_m(\mathcal{F}(\text{clause 9})) = \text{false} . \]
Using the convention that a set of literals represents the universal closure of the disjunction of its elements, clause 9 will be feasible if
\[ \forall \mathcal{M} \left( \neg (\mathcal{F}(\text{clause } 9)) \right) = \text{true}, \]
i.e. if
\[ \forall \mathcal{M} \left( \neg \left( \forall z \forall u \left( \neg D(a,b) \lor \neg P(a) \lor \neg M(b,z,u) \lor \neg D(a,u) \right) \right) \right) = \text{true} \]
which, when the negation sign is distributed and the result is written in the language of the model, gives
\[ \forall \mathcal{M} (\exists z \exists u \left( \{ \{ \mathcal{P}_1 \} \subseteq \{ \mathcal{P}_1, \mathcal{P}_2 \} \} \right) \]
\[ \wedge (\{ \mathcal{P}_1 \} \text{ is a singleton set}) \]
\[ \wedge (\{ \mathcal{P}_1, \mathcal{P}_2 \} \cup z = u) \]
\[ \wedge (\{ \mathcal{P}_1 \} \subseteq u) \right) = \text{true} \].

Clearly there do exist values for \( z \) and \( u \) which make all four of the literals in the above expression true. Thus clause 9 is feasible.

The \( \langle \mathcal{M} \rangle \)-statement for clause 9 is
\[ \forall z \forall u \left( \left( \{ \mathcal{P}_1 \} \subseteq \{ \mathcal{P}_1, \mathcal{P}_2 \} \right) \right) \]
\[ \wedge (\{ \mathcal{P}_1 \} \text{ is a singleton set}) \]
\[ \wedge (\{ \mathcal{P}_1, \mathcal{P}_2 \} \cup z = u) \]
\[ \wedge (\{ \mathcal{P}_1 \} \subseteq u) \right) \]
\[ \longrightarrow \left( \{ \mathcal{P}_1 \} \text{ is not a singleton set} \right) \]
\[ \left. \vee (\{ \mathcal{P}_1, \mathcal{P}_2 \} \cup z \not\subseteq u) \right) \]
\[ \vee (\{ \mathcal{P}_1 \} \not\subseteq u) \]
\[ \vee (\{ \mathcal{P}_1 \} \subseteq z) \right) \].
This is not a true statement (e.g., the valuation $z = \{p_2\}$ and $u = \{p_i, p_2\}$ makes the antecedent true and the consequent false).

Next we form the $\beta_m$-statement for clause 9,

$$
\forall z \forall u \left[ \left( \left( \{p_1\} \subseteq \{p_i, p_2\} \right) \\
\land \left( \{p_i\} \text{ is a singleton set} \right) \\
\land \left( \{p_i, p_2\} \cup z = u \right) \\
\land \left( \{p_i\} \subseteq u \right) \right) \\
\implies \left( \left( \{p_i\} \text{ is a singleton set} \right) \\
\land \left( \{p_i, p_2\} \cup z = u \right) \\
\land \left( \{p_i\} \subseteq u \right) \\
\land \left( \{p_i\} \not\subseteq z \right) \right] .
$$

This statement also is not true (e.g., the valuation $z = \{p_1, p_2\}$ and $u = \{p_i, p_2\}$ makes the antecedent true and the consequent false). Thus $t.v.d. (\text{clause 9, } \beta_m) = T/F$.

We now look at some of the other particular deductions in Figure 2--2. The first clause produced at level 3 has a standard literal set identical to that of clause 8, except for lock numbering. Since it is a unit clause, however, the different lock numbering makes no difference in how the clause can be used. In addition the FSL of this first clause at level 3

$$
\neg D(a,b), \neg M(a,y,b), \neg M(y,a,b)
$$

is completely equivalent to the FSL of clause 8

$$
\neg D(a,b), \neg M(a,y,b)
$$
since our model treats "M" as a predicate which is symmetric on its first two term positions. Thus the first clause at level 3 is marked as an "effective duplicate of clause 8", and is not used in producing any other resolvent. There are 4 more instances of clauses being generated which are, for the purposes of search completeness, duplicates of previously generated clauses. Three of these are exact duplicates of previous clauses, and are so marked in Figure 2--2. This thesis does not treat the issue of subsumption for HLR, since this has not yet been fully explored.

There are also 5 clauses generated which are infeasible. These are listed and marked infeasible in Figure 2--2. All 5 are infeasible for the same reason: they all have both "~D(a,b)" and "D(a,b)" in their FSL's, and thus these FSL's cannot be false in \( \mathcal{M} \).

The search listing terminates with clause 25 which is a null clause. It is required in HLR that the null clause also be feasible, and it is easy to verify that clause 25 is feasible (the FSL consists only of ground literals, and each literal is false when translated into the language of the model \( \mathcal{M} \)). For practical purposes, if an infeasible null clause is produced, this does mean that the input set is unsatisfiable and the search can be terminated. However, such a refutation is not an HLR-refutation.
From Figure 2-2 we have the following number of clauses at each level.

<table>
<thead>
<tr>
<th>level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>number at that level</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>cumulative number up to that level</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>17</td>
<td>20</td>
<td>24</td>
</tr>
</tbody>
</table>

In (Luckham, 1968) this clause set is used with the model partition strategy under two distinct trivial models, neither of which are equivalent to the model used here. For both model partitions the number of clauses in the search space is greater than that obtained here with NLP.

This concludes the treatment of Example II.
2.1.5 The Examples vs. the Formal Definition of HLR.

Except for the lack of factoring in the two examples, the search spaces indicated are those that would be obtained by applying HLR according to the definitions to be given in section 2.4. The definitions in section 2.4 however are slightly different from those given in the examples in the following ways and for the following reasons:

1. the examples were to give an overall intuitive understanding of what HLR is like, and were not intended to define the strategy precisely;
2. the definitions in section 2.4 are intended both to define the HLR refinement unambiguously and to establish a basis strong enough for a relatively simple completeness proof (and this requires that factoring be included);
3. the definitions in section 2.4 define a refinement which allows a more general notion of what a model is than was illustrated in the examples.

The reader should return to Example II after section 2.4 has been assimilated, and at that time Example II can be better understood as a special case of HLR-resolution.

The next two sections deal with the combination issue and the history tracing issue separately, and offer further intuitive insight into why HLR is defined the way it is.
2.2 Combining LR and TMS: The Combination Issue.

Lock resolution (LR) and the model strategy (TMS) are each refutation complete refinements. By LR is meant the strategy of (Boyer, 1971), but without forced merging (see clause 5 in the example below), and with factoring built into the resolution step. TMS is to be the refinement of unrestricted resolution that requires at least one false parent in each resolution step, and has factoring as part of the resolution step, and does not have forced merging (i.e. the clauses may have multiple occurrences of the same literal). The truth values of clauses and of ground literals are determined by some arbitrarily chosen but fixed Herbrand interpretation for the language of the clauses.

A refinement, denoted by $LR \circ TMS$, which requires that all deductions be simultaneously both TMS deductions and LR deductions, can be shown to be incomplete by rather simple examples. For example, the locked sentential clause set

1. $B^3, A^+$
2. $\neg A', \neg B^a$
3. $\neg A^7, B^g$
4. $\neg B^c, A^6$

is unsatisfiable, but there is no LR refutation of this set which is also a TMS refutation using the model $M = \{A, B\}$. However, by adding a simple restriction on the way lock numbers are assigned to literals, a refutation complete
refinement does result. The reader can consult (Loveland, 1978) where an almost identical combination of LR and TMS is considered, called $O_{TM}$-resolution.

Let $S$ be a set of sentential or ground clauses such that each literal in $S$ has a single lock number on it. Then $S$ is said to have an HL-proper lock numbering relative to the model $\mathcal{M}$ iff

1. $\mathcal{M}$ is a Herbrand interpretation for the language of $S$;
2. the lock number of every true literal (according to $\mathcal{M}$) in $S$ is smaller than the lock number of every false literal in $S$.

An example of an HL-proper lock numbering relative to $\mathcal{M} = \{A, B\}$ for the sentential set above is:

1. $B^3, A^4$; true
2. $\neg A^5, \neg B^6$; false
3. $\neg A^7, B^8$; true
4. $\neg B^9, A^6$; true

The following is an LR $\cap$ TMS refutation of this clause set:

2fx4t = 5. $\neg B^2, \neg B^6$; false
5fx1t = 6. $A^6$; true
5fx3t = 7. $\neg A^7$; false
7fx6t = 8. $\Box$; false

where the notation is similar in meaning to that used previously for MLR examples.
The question arises as to whether or not $LR \cap TMS$ is refutation complete if the lock numbering is HL-proper. The answer is that it is complete, and in fact an even stronger result is available, and is stated in the following theorem.

**Theorem 2.2 — 1**

*(Ground LR Refines Ground TMS)*

Let $S$ be a ground set of lock clauses in the language $L$, and $h$ a Herbrand interpretation for $L$. Let $D$ be any $LR$ deduction from $S$. If the lock numbering of $S$ is HL-proper relative to $h$, then $D$ is a TMS deduction from $S$ (using $h$).

**proof:** By induction on the depth of $D$. We number the levels of deduction trees so that the leaf nodes are level zero. The deduction trees are strict binary trees since factoring is done as part of the resolution inference steps. Note that the closure of $S$ under $LR$ is HL-properly lock numbered relative to $h$ if $S$ is.

**base:** depth($D$) = 0. Then $D$ consists of just one clause of $S$. This is also a TMS deduction from $S$ using $h$.

**step:** Assume the theorem true for all $LR$ deductions of depth less than $N$, $N > 0$, and that depth($D$) = $N$. Let $D_1$ and $D_2$ be the deduction trees of the parents of the root of $D$, i.e.
where $R$ is the root of $D$. Let $R_1$, $R_2$ be the roots of $D_1$, $D_2$, respectively. By the induction hypothesis $D_1$ and $D_2$ are each TMS deductions from $S$ using $h$. By the theorem hypothesis $R_1 \cdot R_2 = R$ is an LR inference step, and $R_1$ and $R_2$ have an HL-proper lock numbering relative to $h$. If both $R_1$ and $R_2$ were false clauses, then there could be no complementary literals on which to resolve to produce $R$. Thus at least one of $R_1$, $R_2$ is true. Without loss of generality we can assume $R_1$ is true. Therefore $R_1$ has at least one literal true in $h$. But the lock numbering is HL-proper, so that the literal(s) resolved on in $R_1$ to produce $R$ is (are) true in $h$. Therefore the literal(s) resolved on in $R_2$ is (are) false in $h$. But then a lowest lock numbered literal in $R_2$ is false. But the lock numbering is HL-proper, therefore all literals in $R_2$ are false. Therefore $R_2$ is a false clause. Thus $R_1 \cdot R_2 = R$ is a TMS inference step using $h$. Therefore $D$ is a TMS deduction from $S$ using $h$.

Q.E.D.
Corollary 2.2-2

(Ground Completeness of LR $\cap$ TMS)

Let $S$ be an unsatisfiable set of ground clauses in the language $L$, and $h$ be a Herbrand interpretation for $L$. Then if an HL-proper lock numbering (relative to $h$) is given to $S$, there exists an LR $\cap$ TMS refutation of $S$.

proof: LR is refutation complete for an arbitrary lock numbering; therefore there exists an LR refutation of $S$ when the lock numbering is HL-proper relative to $h$. By Theorem 2.2-1 this refutation is also a TMS refutation using $h$, and thus it is an LR $\cap$ TMS refutation.

Q.E.D.

By definition an LR $\cap$ TMS refutation is a TMS refutation, so that Corollary 2.2-2 includes the completeness of TMS at the ground level also. The following theorem is the completeness theorem for TMS (see the discussion in section 1.2 as to why we attribute TMS to (Luckham, 1970)).
Theorem 2.2 -- 3

(Completeness of TMS)

Let $S$ be an unsatisfiable set of clauses, and $h$ a Herbrand interpretation for the language of $S$. Then there exists a TMS refutation of $S$ using $h$.

**proof:** Use Herbrand's theorem to assert the existence of an unsatisfiable set of ground instances of clauses in $S$. By the refutation completeness of ground TMS there exists a TMS refutation, $R$, of this set, using $h$. For each triple of node clauses, $c_1$, $c_2$, $c_3$ such that $c_1$ and $c_2$ resolve to produce $c_3$ in $R$, exactly one of $c_1$ and $c_2$ is false in $h$. $R$ is also an unrestricted resolution refutation, and can be lifted to a general refutation of $S$ in the usual manner (Chang and Lee, 1973). The lifted tree is also a TMS refutation of $S$ using $h$, since each false clause lifts to a false clause, and each true clause lifts to either a true or false clause.

Q.E.D.

It has been shown that the notion of an HL-proper lock numbering in LR established a connection between LR and TMS at the ground level. Theorem 2.2-1 shows that LR with HL-proper lock numberings actually is a refinement of TMS at the ground level.
This connection does not exist in the same form at the general level however. The reason is that, for the types of models we are interested in, a literal containing variables has no single truth value, but rather has a truth value that is a function of its variables. Assigning single lock numbers to literals in a general clause does not ensure that all ground instances of the clause have an HL-proper lock numbering. However, the assignment of two lock numbers (from two non-overlapping intervals of integers) to each general literal ensures that every ground instance of every clause can have an HL-proper lock numbering by choosing, for each specific ground instance, which of the two lock numbers is "active" on each ground literal. This is in fact done in HLR in the sense that it is implicit in the inference rule in the way the FSL's accumulate literals and are tested for feasibility.

The previous paragraph contains the critical concept around which the HLR refinement is structured, i.e. to deal sharply at the general level, on a semantic basis, with clauses in which the truth values of literals differ for different ground instances of the clause. Because of this situation (of semantically distinguishable ground instances) semantic refinements which have been proven complete at the ground level usually present difficulties in being lifted in a form as strong as the ground level completeness allows. Several times in the resolution literature there have been confrontations with and attempts to deal with this
situation, and we mention here three mechanisms that have been used.

1. The assignment of a truth value "false" to a clause which has at least one ground instance false in the chosen Herbrand interpretation. In TMS this leads to the possibility of general level resolution steps between two clauses which are both "false", although at the ground level this is an impossibility. Likewise in semantic resolution (Slagle, 1967) and LI - rz resolution (Slagle, 1972), both of which are latent clash rules, the nucleus is necessarily true at the ground level, but is allowed to be "false" at the general level. This leads to general level strategies which are not ground faithful (see section 5.1).

2. The requirement of a decomposability property, as defined in (Brown, 1973). Brown deals with the use of several models simultaneously for Horn sets, but encounters exactly the same problem under discussion here. Brown does not fully present effective methods for decomposing clauses so that the decomposability property holds. In (Sandford, 1977a) a form of HLR, called the primitive representation form, is introduced which is an effective decomposition method.

3. The use of restricted types of Herbrand interpretations (trivial models) or closely related notions (homogeneous settings (Loveland, 1978)) which
assign a truth value of "false" to a literal whenever that literal has an instance that is "false".

For most clause sets the type 3 mechanisms are simply unable to model adequately the semantic subtleties which are both relevant to the intended domain of interpretation and encompassed by the syntactic complexity of the clauses. The type 1 mechanism allows easy lifting, e.g. in the completeness proof of TMS (Theorem 2.2-3), but also allows general level deductions which correspond to no ground level deductions which are in the same refinement, and thus increases the number of clauses deducible beyond that necessary for completeness (this will be illustrated in section 2.3 for TMS).

The HLR refinement presented in this thesis essentially uses the type 2 mechanism for lifting, but does so in a somewhat obscure manner which will not be elaborated upon here. For the reader who is studying HLR with a view towards either implementation, or further research into extending HLR, it is strongly recommended that the primitive representation form of HLR in (Sandford, 1977a) be studied also.
2.3 HLR and History Tracing.

HLR uses the FSL lists to accumulate certain relevant information about how a clause was derived. A similar thing is applicable to TMS. In this section we define a refinement, denoted by TMS + FSL, give an example showing that it can eliminate from a search space some clauses that TMS cannot, and prove its refutation completeness.

The refinement TMS + FSL is essentially HLR without lock numbers, and is defined as follows.

1. Clauses are comprised of standard literals and FSL literals, as in HLR, but there are no lock numbers.
2. The model to be used is an arbitrarily chosen Herbrand interpretation for the language of the clauses.
3. The notions of semantic function, feasibility, $\alpha$- and $\beta$-statements for a clause, and t.v.d. values are all identical to what they are for HLR.
4. A normal set of input clauses is transformed into a TMS + FSL input set by adding an empty FSL to each clause.
5. Each resolution step identifies one parent as the true parent and the other as the false parent. The t.v.d. of the true parent must be T or T/F, and the t.v.d. of the false parent must be F or T/F. The inference step is asymmetrical with respect to the two parents, and we say the false parent is resolved against the true parent.

6. In resolving C\textsubscript{i} against C\textsubscript{j}, the resolvent, C\textsubscript{k}, is formed as follows:
   
i. the standard literals of C\textsubscript{k} are the result of an unrestricted resolution step done on the standard literals of C\textsubscript{i} and C\textsubscript{j} (factoring is part of the resolution step);
   
ii. the FSL set of C\textsubscript{k} is the set union of the FSL literals of C\textsubscript{i} and C\textsubscript{j} and the standard literals of C\textsubscript{i} (the resulting FSL of C\textsubscript{k} has the resolution unifier applied to its literals also).

7. Only feasible clauses are kept.

8. A TMS + FSL refutation of S is a TMS + FSL deduction from S of a feasible clause containing no standard literals.

An example of TMS with FSL's is given in Figure 2--3, which is the clause set of Example 1 of section 2.1, and uses the same model. The complete search space is listed and is seen to have only the 2 clauses of level 1 as the retained clauses.
\[ \mathcal{M} = \{ \neg P(a), P(b), \neg Q, R, S \} \]

1. \( \neg P(x), Q(x) \); \( FSL = \{\} \) T/F
2. \( \neg Q(x), R(x) \); \( FSL = \{\} \) T
3. \( \neg R(a) \); \( FSL = \{\} \) F
4. \( P(x), S(x, a, b) \); \( FSL = \{\} \) T

**Figure 2 → 3 ATMS + FSL Search**

\[ 1Fx2T = 5. \quad \neg P(x), R(x); \quad FSL = \{\neg P(x), Q(x)\} \quad T \]
\[ 3Fx2T = 6. \quad \neg Q(a); \quad FSL = \{\neg R(a)\} \quad T \]
\[ 1Fx4T = 7. \quad Q(x), S(x, a, b); \quad FSL = \{\neg P(x), Q(x)\} \quad T \]

**Figure 2 → 4 AN INFEASIBLE DEDUCTION**

\[ 3Fx5T = \text{INFEASIBLE}. \quad \neg P(a); \quad FSL = \{\neg P(a), Q(a), \neg R(a)\} \]
\[ 1Fx6T = \text{INFEASIBLE}. \quad \neg P(a); \quad FSL = \{\neg R(a), \neg P(a), Q(a)\} \]
Figure 2--4 gives the deduction tree of the first clause of level 2, and illustrates the reason why this clause is infeasible. The reason is that the resolution step resolving clause 1 against clause 2 must use clause 1 as the false parent, and this is recorded in the FSL of clause 5 by putting the standard literals of clause 1 into it. When clauses 5 and 3 resolve the unifier is such that clause 1 must represent, in this deduction, the ground clause "¬P(a), Q(a); FSL = {}" which is a true clause in our model. This causes the FSL of the resolvent of clauses 3 and 5 to be true, and thus the resolvent is infeasible.

The other level 2 clause is infeasible for essentially the same reason. It should now be intuitively clear what FSL's do in TMS + FSL; they prevent searching over general level deduction trees which cannot possibly represent any ground TMS deduction. This gives TMS + FSL a characteristic we call ground faithfulness, which is defined in section 5.1. The FSL's play much the same role in HLR, although there the situation is more complex and less intuitive.

It is seen from this example that the use of FSL literals has prevented all of the deductions of the unit clause "P(a)", which was derived twice by TMS (see Figure 1--1). Thus the use of FSL's does increase the deduction incompleteness of TMS, and we are led to ask: Is TMS + FSL refutation complete?
Theorem 2.3 -- 1

(Ground Completeness of TMS + FSL)

Let \( G \) be an unsatisfiable set of ground normal clauses, and \( h \) be a Herbrand interpretation for a language of \( G \). Let \( G^* \) be the TMS + FSL input set for \( G \). Then there exists a TMS + FSL refutation of \( G^* \) using \( h \).

proof: By the completeness of ground TMS, there exists a TMS refutation, \( R \), of \( G \), using \( h \). Let the depth of \( R \) be \( N \). In \( R \) every internal node has a parent true in \( h \) and a parent false in \( h \). We transform \( R \) to the tree \( R^* \) as follows. Add an empty FSL to each leaf of \( R \); the original literals at the leaf are now standard literals. In order, for \( i = 1, 2, \ldots, N \), for each clause \( g \) at level \( i \), add to \( g \) an FSL containing all of the FSL literals of its two parents, plus all of the standard literals in the parent of \( g \) which is false in \( h \). The original literals at \( g \) are now standard literals.

In \( R^* \) every literal in every FSL is a false ground literal. Therefore every clause in \( R^* \) is feasible. Let \( g \) be an arbitrary clause in \( R^* \). If the standard literals of \( g \) are all false in \( h \), then, since the FSL of \( g \) contains only false ground literals (or is empty), it is easy to verify that the \( \beta \)-statement of \( g \) is true and the \( \alpha \)-statement is false, so that the t.v.d. of \( g \) is \( F \). Similarly if the standard literals of \( g \) contain one or more true literals, then the t.v.d. of \( g \) is \( T \).
Now let \( g \) be some internal node in \( R^* \), and \( g_1 \) be its parent with all false standard literals, and \( g_2 \) its other parent. By the construction of \( R^* \), \( g \) is a TMS + FSL resolvent of \( g_1 \) against \( g_2 \), specifically:

1. the t.v.d. values are correct so that \( g_1 \) can be the false parent and \( g_2 \) the true parent;
2. \( g, g_1, g_2 \) are all feasible;
3. the standard literals of \( g \) have been formed by a TMS inference step on the standard literals of \( g_1 \) and \( g_2 \), which qualifies as a special case of an unrestricted resolution step.

Therefore each immediate deduction in \( R^* \) is a TMS + FSL deduction. The leaves of \( R^* \) are all clauses in \( G^* \). The root of \( R^* \) is a feasible clause with no standard literals. Therefore \( R^* \) is a TMS + FSL refutation of \( G^* \).

Q.E.D.

Theorem 2.3 -- 2

(Completeness of TMS + FSL)

Let \( C \) be an unsatisfiable set of normal clauses, and \( h \) be a Herbrand interpretation for the language of \( G \). Let \( G^* \) be a TMS + FSL input set for \( C \). Then there exists a TMS + FSL refutation of \( G^* \) using \( h \).
proof: By Herbrand's theorem there exists a finite unsatisfiable set of ground clauses, $\hat{G}$, each of which is an instance of a clause in $G$. Let $\hat{G}^*$ be the TMS + FSL input set for $\hat{G}$. Each clause in $\hat{G}^*$ is a ground instance of a clause in $G^*$. By Theorem 2.3-1 there exists a TMS + FSL refutation, $\hat{R}$, of $\hat{G}^*$, using $h$. It should be clear that the usual lifting lemma (see, e.g. (Chang and Lee, 1973)) can be modified so that it "lifts" pairs of sets of literals. By the use of such a modified lifting lemma, and the fact that the leaves of $\hat{R}$ are instances of clauses in $G^*$, we can assert the existence of a tree $R$ with the following properties:

i. the leaves of $R$ are clauses in $G^*$;
ii. $\hat{R}$ and $R$ are tree isomorphic under $\Lambda$, i.e. $\Lambda$ is a total function on nodes of $\hat{R}$, is 1-1 onto $R$, and for every $g$ a node of $\hat{R}$ there exists a substitution $\sigma^g$ such that $(\Lambda(g))\sigma^g = g$, and $\Lambda$ preserves the parenthood relation;
iii. if $\hat{C}_i$ resolves against $\hat{C}_j$ to produce $\hat{C}_k$ in $\hat{R}$, then $\Lambda(C_i)$ resolves against $\Lambda(C_j)$ to produce $\Lambda(C_k)$ in $R$ by the TMS + FSL inference rule, assuming the t.v.d. and feasibility conditions are satisfied.

Thus, if it can be shown that the t.v.d. and feasibility conditions are met for each immediate deduction in $R$, then $R$ will be a TMS + FSL refutation of $G^*$ using $h$. We first notice that every clause in $R$ is feasible, since there is a ground instance (in $\hat{R}$) of each clause which has only false
literals in its FSL.

Now let \( \hat{C}_k \) be any internal node in \( \hat{N} \), and \( \hat{C}_i \) its false parent and \( \hat{C}_j \) its true parent. Let \( C_k, C_i, C_j \) be the corresponding \((\land)\) clauses in \( \mathcal{K} \). There exists a substitution, \( \sigma \), such that \((C_i)\sigma = \hat{C}_i\), and
\[
\sigma = \{ t_1/x_1, \ldots, t_m/x_m \}
\]
where \( t_1, \ldots, t_m \) are ground terms, and \( x_1, \ldots, x_m \) are all of the distinct variable symbols appearing in (standard and FSL) literals in \( C_i \). Since \( \hat{C}_i \) is the false parent in a TMS + FSL inference step, and it is ground, it is the case that, in the interpretation \( h \),
\[
\neg (\forall (\mathcal{F}(\hat{C}_i))) \text{ is true}
\]
and
\[
(\forall (\mathcal{S}(\hat{C}_i))) \text{ is false.}
\]
Thus
\[
\neg (\forall (\mathcal{F}((C_i)\sigma))) \text{ is true}
\]
and
\[
(\forall (\mathcal{S}((C_i)\sigma))) \text{ is false.}
\]
Therefore the \( \alpha \)-statement of \( C_i \),
\[
\forall x_1 \ldots x_m [ \neg (\forall (\mathcal{F}(C_i))) \land (\forall (\mathcal{S}(C_i))) ]
\]
is false, since for the valuation
\[
x_1 = t_1, \ldots, x_m = t_m
\]
the antecedent is true and the consequent is false. Therefore the t.v.d. of \( C_i \) is \( T \) or \( F \), depending upon whether the \( \beta \)-statement of \( C_i \) is true or not. In a similar manner the t.v.d. of \( C_j \) can be shown to be \( T \) or \( F \).
Therefore Ci can be the false parent, and Cj the true parent, in a TMS + FSL inference step which results in Ck. But Ck was arbitrarily chosen as an internal node in R. Thus every internal node in R results from a TMS + FSL inference step. Therefore R is a TMS + FSL refutation of C* using h.

Q.E.D.

Now that the combination and history tracing issues have been dealt with separately the reader should have an intuitive feel for their role in determining the nature of the HLR inference rule as presented in the examples of section 2.1.

We mention here that semantic P-deduction (Slagle, 1967) and positive-unit semantic resolution for Horn sets (Henschel, 1975) can have FSL's added in the obvious way. For example in semantic P-deduction the resolvent of a latent maximal semantic P-clash will have an FSL containing all of the following literals:

i. all of the FSL literals in the nucleus and the electrons;

ii. all standard literals in all electrons;

iii. all standard literals in the nucleus which are not resolved with electron literals.

The model is any Herbrand interpretation for the language of the clauses. Input clauses have empty FSL's attached to
them, and all retained clauses are required to be feasible. The completeness of such a combination of FSL's and semantic P-deduction is easily proven by the same type of proof as used for TMS + FSL (Theorems 2.3-1 and 2.3-2). A similar thing can be done for Meuschen's semantic Horn refinement.

In the next section a rigorous definition of HLR is given which deals with the combination and history tracing aspects of the refinement, along with the model scheme issue. No explicit mention of model schemes is involved, however, since the definitions incorporate explicitly only the semantic functions of models schemes. The model schemes themselves will be dealt with in Chapter 4.
2.4 Formal Definition of NL-Resolution.

NL-resolution will be defined using a simple and obvious notation, where, for example a statement such as

$$\forall x \forall z \exists y: (x, y) \text{ and } (y, z)$$

means "for all $x$ and $z$ there exists a $y$ such that the ordered pair $(x, y)$ is in the relation $R_1$ and the pair $(y, z)$ is in the relation $R_2$". Unbound variables are to be understood as universally quantified, unless the immediate context indicates otherwise. Certain symbols will be generic or typed symbols which are variables restricted to objects of only one type.

The definitions are written using ordinary set theory notation. Occasionally it is more convenient to think of the sets as multi-sets. In general multi-sets are intuitively clearer for thinking about ground instances of clauses, and they allow arguments to be based on the notion of direct instances (Boyer, 1971). The definitions of this section are intended primarily to hold true for ordinary sets but are usually also applicable to multi-sets and lists.

Sections 2.4.1 through 2.4.22 are all definitions, and constitute a relatively complete specification of NLR. Some of the definitions are of standard concepts in resolution theory or logic, and are included for purposes of exposition. Any undefined notions from resolution are to be taken as defined in (Chang and Lee, 1973).
2.4.1 Notation. We assume that a first order language, \( L \), has been defined in the usual way (e.g. (Kleene, 1967)) so that there is a well defined set of literals for the language. The usual notions of literal, clause (i.e. a set of literals) and set of clauses in resolution will be referred to as normal literal, normal clause and normal set of clauses, and denoted by the generic symbols, \( N, \overline{N}, \lor \), respectively. The corresponding notation for HL-literal, etc. (defined below), is \( \overline{H}, \overline{\overline{H}}, \overline{\lor} \).

2.4.2 HL-Literal. An HL-literal, \( H \), hereafter referred to as simply a literal, is an ordered triple \( H = \langle i, N, j \rangle \) where \( N \) is a normal literal and \( i \) and \( j \) are integers. The selector functions for the components are defined so that \( H = \langle t(H), l(H), f(H) \rangle \) where \( t(H) \) and \( f(H) \) are the true and false lock numbers of \( H \), and \( l(H) \) is the literal core of \( H \). Two literals are equal iff they are identical in all three components.

Let \( K_i \) be either a literal or a set of literals, for \( i = 1, 2, \ldots, n \). We extend the domain of \( l \) as follows:

\[ 1(\{K_1, K_2, \ldots, K_n\}) = \{ l(K_1), l(K_2), \ldots, l(K_n) \} \]

The negation operator applied to a literal is defined by:

\[ \overline{H} = \langle t(H), \overline{l(H)}, f(H) \rangle \]

where the string "\( \overline{\overline{\overline{}}} \)" is replaced by the null string.
2.4.3 \textbf{HL-Clause}. An HL-clause, \( \mathcal{H} \), hereafter simply referred to as a \textit{clause}, is an ordered pair of sets of literals:

\[ \mathcal{H} = \langle \mathcal{S}(\mathcal{H}), \mathcal{F}(\mathcal{H}) \rangle \]

where \( \mathcal{S} \) and \( \mathcal{F} \) are selector functions. \( \mathcal{S}(\mathcal{H}) \) is the set of \textbf{Standard literals} of \( \mathcal{H} \), and \( \mathcal{F}(\mathcal{H}) \) is the \textbf{False Substitution List (FSL)} of \( \mathcal{H} \).

\( \mathcal{S} \) is extended to sets of clauses, \( \mathcal{H} \), by

\[ \mathcal{S}(\mathcal{H}) = \{ \mathcal{S}(\mathcal{H}) \mid \mathcal{H} \in \mathcal{H} \} \]

Note that \( \mathcal{S}(\mathcal{H}) \) is a normal set of clauses.

When writing clauses for examples we use the form previously used (e.g., in Figure 2--2), as this is much more readable.

2.4.4 \textbf{Unconstrained}.

2.4.4.1 (unconstrained \( \emptyset \)) iff \( (\mathcal{F}(\emptyset) = \{\}) \).

2.4.4.2 (unconstrained \( \mathcal{H} \)) iff \( (\forall \mathcal{H}: (\mathcal{H} \in \mathcal{H}) \implies (\text{unconstrained } \mathcal{H})) \).

The first 7 clauses in Figure 2--2 comprise an unconstrained clause set.

2.4.5 \textbf{Languages}. Let \( L \) be any first order language and let \( \mathcal{N} \) be any (finite) set of normal literals of \( L \). We associate with \( \mathcal{N} \) a set of formulas of \( L \) each of which is obtained by taking the universal closure of a disjunction of all the literals of \( \mathcal{N} \). We use \( \mathcal{N} \) ambiguously to refer to the set of literals and to any of the formulas associated with the set.
in the way described. Similarly, let \( \mathcal{N} \) denote any of the formulas obtained by taking the existential closure of a conjunction of the complements of literals in \( \mathcal{N} \). Thus we can use the notation \( (\mathcal{N} \in L) \). We use, interchangeably, the phrases "a language of" and "a language for", and define:

2.4.5.1 \((L a language of \mathcal{N})\) iff
\[ (\forall \mathcal{N} : \mathcal{N} \in \mathcal{N} \implies \mathcal{N} \in L) \.
\]

2.4.5.2 \((L a language of \mathcal{H})\) iff
\[ \{ (1(\mathcal{L}(H)) \in L) \mid (1(\mathcal{F}(H)) \in L) \} \]

N.B. \( L a language of \mathcal{H} \) does not mean \( \mathcal{H} \in L \).

2.4.5.3 \((L a language for \mathcal{H})\) iff
\[ (\forall \mathcal{H} : \mathcal{H} \in \mathcal{H} \implies (L a language for \mathcal{H}) \).
\]

2.4.5.4 For \( K \) one of \( \mathcal{H}, or \mathcal{N} \), define
\((L the language of K)\) iff
\[ (L = \bigcap \{ L_{\mathcal{K}} \mid (L_{\mathcal{K}} a language for K) \} \).
\]

2.4.5.5 \((s a sentence of L)\) iff
\[ \{ (s \in L) \mid (s contains no free variables) \} \]

2.4.6 **Substitutions, Unifiers, Ground Instances.**

\( \sigma, \rho \) and \( \tau \) will be generic symbols for substitutions.

2.4.6.1 We write
\[ (\sigma a substitution of L) \]
if \( \sigma \) is a finite set of elements, each a string of the form \( t/v \), where \( t \) is a term of \( L \) distinct from \( v \), and \( v \) is a variable symbol of \( L \) and no two elements of \( \sigma \) have the same variable symbol after the stroke symbol. In the usual way we consider each \( \sigma \) to define a function from strings over
the alphabet of \( L \) into strings over the alphabet of \( L \). If \( J \) is a string over the alphabet of \( L \), then we denote the application of \( \sigma \) to \( J \) by \( J\sigma \), and say that \( J\sigma \) is an instance of \( J \). Sometimes \( J\sigma \) is written as \((J)\sigma\).

2.4.6.2 We define

\[ H\sigma = \langle t(H), (1(H))\sigma, f(H) \rangle. \]

For \( K \) a set of literals, normal literals, or lock literals,

\[ K\sigma = \{ k\sigma \mid k \in K \}. \]

For a clause, \( \mathcal{H} \)

\[ \mathcal{H}\sigma = \langle (\mathcal{S}(\mathcal{H}))\sigma, (\mathcal{F}(\mathcal{H}))\sigma \rangle. \]

2.4.6.3 (ground \( \mathcal{H} \)) iff (\( \mathcal{H} \) contains no variable symbols).

(ground \( \mathcal{H} \)) iff (ground \( 1(\mathcal{H}) \)).

For \( K \) a set of (normal or NL-) literals:

(ground \( K \)) iff \( [ \forall k: k \in K \implies (\text{ground } k) ] \).

(ground \( \mathcal{H} \)) iff \( [ \text{ground } \mathcal{S}(\mathcal{H}) \text{ (ground } \mathcal{F}(\mathcal{H}) \) ] \).

2.4.6.4 For \( K \) a (normal or NL-) literal or set of literals,

(\( \sigma \)-grounds \( K \)) iff (ground \( K\sigma \)).

(\( \sigma \)-grounds \( \mathcal{H} \)) iff (ground \( \mathcal{H}\sigma \)).

2.4.6.5 If \( K \) is a set of literals, then

(\( \text{unifiable } K \))

iff

[ \exists \sigma: (1(K\sigma) \text{ is a singleton set}) ].

We use "\text{mgu}" as a function mapping sets of unifiable literals to the most general unifier of the set (Robinson, 1965).
2.4.7 Semantic Functions. $\mathcal{M}$ is the generic symbol for models. There is as yet no commitment as to the nature of the model. In the particular case where $\mathcal{M}$ is a Herbrand interpretation, $h$, we write this as $\mathcal{M}_h$.

2.4.7.1 We write

$$(h \text{ a Herbrand interpretation for } L)$$

with the obvious meaning intended (see section 1.2).

$\psi$ is the generic symbol for semantic functions.

2.4.7.2 ($\psi$ a semantic function for $L$) iff

$$(\psi \text{ is a total function on the set of sentences of } L \text{ into } \{\text{true, false} \} ).$$

We assume that each model, $\mathcal{M}_h$, defines a semantic function, $\psi_h$, called the associated semantic function for $\mathcal{M}_h$. The HLR refinement depends only upon $\psi_h$, and not upon the internal mechanisms by which $\mathcal{M}_h$ determines $\psi_h$.

2.4.8 The Herbrand Semantic Function.

If

$$(h \text{ a Herbrand interpretation for } L)$$

then we use the specific notation $\phi_h$ for the associated semantic function for $\mathcal{M}_h$, where $\mathcal{M}_h = h$. $\phi_h$ is defined in the usual way as follows:

2.4.8.1 If (ground $N$) then

$$\phi_h(N) = \begin{cases} 
\text{true} & \text{if } N \in \mathcal{M}_h \\
\text{false} & \text{otherwise}
\end{cases}$$
2.4.8.2 If (ground $\mathcal{N}$) then

$$
\phi_\lambda(\mathcal{N}) = \begin{cases} 
\text{true} & \text{if } \mathcal{N} \cap \mathcal{M}_\lambda \neq \emptyset \\
\text{false} & \text{otherwise}
\end{cases}
$$

2.4.8.3 For not necessarily ground $\mathcal{N}, \mathcal{N}'$,

$$
\phi_\lambda(\mathcal{N}) = \begin{cases} 
\text{true} & \text{if } \forall \sigma : \{ (\sigma \text{ grounds } \mathcal{N}) \land (\sigma \text{ a substitution of } \mathcal{L}) \} \longrightarrow (\phi_\lambda(\mathcal{N} \sigma^{-1}) = \text{true}) \\
\text{false} & \text{otherwise}
\end{cases}
$$

$$
\phi_\lambda(\mathcal{N}') = \begin{cases} 
\text{true} & \text{if } \forall \mathcal{N} : (\mathcal{N} \in \mathcal{M} - \mathcal{N} \sigma^{-1}) \longrightarrow \phi_\lambda(\mathcal{N}) = \text{true}) \\
\text{false} & \text{otherwise}
\end{cases}
$$

2.4.8.4 For a ground literal $\lambda$,

$$
\phi_\lambda(\lambda) = \phi_\lambda(l(\lambda)).
$$

In general the lock numbers on literals are invisible for semantic purposes. Thus, for example, in later chapters a set of literals

$$
\{l_1, l_2, \ldots, l_m\}
$$

will be treated, semantically, as if it were the set

$$
\{l(\lambda_1), l(\lambda_2), \ldots, l(\lambda_m)\}.
$$

2.4.8.5 If (unconstrained $\mathcal{N}$) then $\phi_\lambda(\mathcal{N}) = \phi_\lambda(l(S(\mathcal{N})))$.

If (unconstrained $\mathcal{N}$) then $\phi_\lambda(\mathcal{N}) = \phi_\lambda(l(S(\mathcal{N})))$. 
2.4.8.6 For $s$ a sentence of $L$, $\mathbf{a}_h(s)$ is to be the usual truth value of $s$ in the relational structure induced by $h$. The relational structure induced by $h$ is given by:

1. the domain, $\mathcal{U}_h$, is the Herbrand universe for $L$ (a set of strings);

2. each constant symbol, $C$, of $L$, is assigned (i.e. interpreted as) the string $C$ in $\mathcal{U}_h$;

3. each $n$-ary function symbol, $f^r$, $n > 0$, of $L$, is assigned an $n$-ary function over $\mathcal{U}_h$, such that the $n$-tuple $t_1, \ldots, t_n$ of terms in $\mathcal{U}_h$, is mapped to the string "$f^r(t_1, \ldots, t_n)$";

4. each $n$-ary predicate symbol, $P^r$, $n > 0$, of $L$, is assigned an $n$-ary relation on $\mathcal{U}_h$, such that the $n$-tuple $t_1, \ldots, t_n$ is in the relation iff $P^r(t_1, \ldots, t_n) \in h$.

We say a set of sentences, $K$, is true in $h$, i.e. $\mathbf{a}_h(K) = \text{true}$, if for all $k \in K$, $\mathbf{a}_h(k) = \text{true}$.

A normal clause, $\mathcal{N}$, has a truth value according to this definition (when considered as a sentence in $L$) which is the same as in definition 2.4.8.3 (and likewise for a normal set of clauses, $\mathcal{N}$). We need this current definition because we will need to assign truth values to $\alpha$- and $\beta$-statements (which are not simply sets of literals) based on a Herbrand interpretation.
2.4.8.7 Let $A$ and $B$ be clauses, sets of clauses, sentences, or sets of sentences in a language $L$. Then we write $A \models B$ to mean that $B$ is true in every interpretation for $L$ in which $A$ is true (we assume the usual amenities, e.g. a set of sentences is equivalent to the conjunction of its members).

2.4.9 *False Permissiveness.*

Suppose $\psi_1$ and $\psi_2$ are both semantic functions for the same language, $L$. Then we define

\[
(\psi_1 \text{ false permissive wrt. } \psi_2)
\]

iff

\[
(\forall s: (s \text{ a sentence of } L) \implies ((\psi_2(s) = \text{false}) \implies (\psi_1(s) = \text{false}))\).
\]

Notice that "false permissive wrt." is a transitive relation.

2.4.10 *Sound Semantic Functions.*

\[
(\psi \text{ a sound semantic function for } L)
\]

iff

\[
(\forall \phi_1: (h \text{ a Herbrand interpretation for } L) \implies \pi \text{ false permissive wrt. } \phi_1).
\]

When the language is known from context we say simply (sound $\psi$) instead of (\psi a sound semantic function for L).
Clearly we have

\[(h \text{ a Herbrand interpretation for } L) \implies \text{(sound } \phi_h \text{)}.\]

It is also easily shown that

\[\{ (\psi_i \text{ false permissive wrt. } \psi_k) \text{ (sound } \psi_k) \} \implies \text{(sound } \psi_i).\]

2.4.11 Unsatisfiability.

\[(\text{unsatisfiable } \mathcal{N})\]

iff

\[\{ (L \text{ the language of } \mathcal{N}) \implies \]

\[(\forall h: (h \text{ a Herbrand interpretation for } L) \implies (\phi_h(\mathcal{N}) = \text{false}) ) \}.\]

If (unconstrained \( \mathcal{N} \)), then we define

\[(\text{unsatisfiable } \mathcal{N})\]

iff

\[\{ (L \text{ the language of } \mathcal{N}) \implies \]

\[(\forall h: (h \text{ a Herbrand interpretation for } L) \implies (\phi_h(\mathcal{N}) = \text{false}) ) \}.\]

Thus an unconstrained set of clauses, \( \mathcal{N} \), is unsatisfiable iff \( L(\mathcal{I}(\mathcal{N})) \) is unsatisfiable. In the above two definitions "the language of" may be replaced by "a language for" with no substantive change.

2.4.12 Feasibility.

\[(\mathcal{N} \text{-feasible } \mathcal{N})\]

iff

\[(\psi(\mathcal{N}(\mathcal{N})) = \text{false}).\]
This is occasionally shortened to just \((\text{feasible} \mathcal{H})\). When 
\(\psi\) is the associated semantic function of \(\mathcal{M}\), then the 
phrase "\(\mathcal{H}\) is feasible in \(\mathcal{M}\)" will also be used to mean 
"\(\psi\)-feasible \(\mathcal{H}\)".

2.4.12 \(\vdash\) and \(\beta\)-Statements. If \(K\) is a set of literals, 
then \(\lor(K)\) is the disjunction of the elements of \(K\). To be 
completely precise it would be necessary to also state the 
order in which the literals in \(K\) appear in \(\lor(K)\), since 
statements in a language are strings of symbols. None of 
the uses of the "\(\lor\)" function will depend upon the actual 
order in which the literals are written, so we leave this 
order undefined.

Let \(\forall^x\) be \(\forall x_1 \forall x_2 \ldots \forall x_p\), where \(x_1, \ldots, x_p\) are 
all of the distinct variable symbols contained in \(\mathcal{H}\). Then 
we define the \(\vdash\) and \(\beta\)-statements of a clause:

\[
\vdash(\mathcal{H}) \triangleq \forall^x(\lor(1(\exists(\mathcal{H})))) \rightarrow \lor(1(\forall(\mathcal{H})))
\]

\[
\beta(\mathcal{H}) \triangleq \forall^x(\lor(1(\exists(\mathcal{H})))) \rightarrow \lor(1(\forall(\mathcal{H})))
\]

For a clause with an empty FSL, the expression 
\(\lor(1(\exists(\mathcal{H})))\) is the empty conjunction, denoted by \(\Box\).

Notice that 

\[
(l \text{ a language of } \mathcal{H}) \rightarrow \begin{cases} 
\vdash(\mathcal{H}) \text{ a sentence of } l \\
\beta(\mathcal{H}) \text{ a sentence of } l 
\end{cases}
\]
2.4.14 Truth Value Designator (t.v.d.).

Under the following conditions:

(L a language of $\mathcal{H}$)

($\psi$ a semantic function for $L$)

($\psi$-feasible $\mathcal{H}$)

we define the truth value designator (t.v.d.) of a clause $\mathcal{H}$ under the semantic function $\psi$ as:

$$t.v.d.(\mathcal{H}, \psi) = \begin{cases} 
T & \text{if } \left( \psi(\alpha(\mathcal{H})) = \text{true} \right) \left( \psi(\beta(\mathcal{H})) = \text{false} \right) \\
F & \text{if } \left( \psi(\beta(\mathcal{H})) = \text{true} \right) \left( \psi(\alpha(\mathcal{H})) = \text{false} \right) \\
T/F & \text{otherwise}
\end{cases}$$

The distinction between this definition of t.v.d. compared to that used for Examples I and II is that here the definition includes the case of a semantic function mapping both the $\alpha$- and $\beta$-statements of a clause to true. In several proofs of theorems in later chapters it will become clear that the $\alpha$- and $\beta$-statements cannot be simultaneously true when $\psi$ is sound.

2.4.15 Null Clause.

(null $\mathcal{H}$) iff \{ (feasible $\mathcal{H}$) ($\mathcal{S}(\mathcal{H}) = \{}$ ) \}. If (null $\mathcal{H}$) then $\mathcal{H}$ is called a null clause. Notice that the feasibility requirement necessitates that a semantic function be known from context.
2.4.16 **HL-Proper Lock Numbering**.

Let \( \mathcal{H} \) be a set of clauses, where each clause is indexed by an integer and each literal of each clause is indexed by an integer such that
\[
\mathcal{H} = \{ H^x_i \mid 1 \leq i \leq n \},
\]
for \( n \) a non-negative integer, and
\[
H^x_i = \langle \{ H^x_j \mid 1 \leq j \leq \psi_i \}, \mathcal{F}(H^x) \rangle
\]
for \( 1 \leq i \leq n \), and \( \psi_i, \psi_n \ldots \psi_m \) all non-negative integers. Then

\[(\text{HL-proper } \mathcal{H})\]

iff
\[
[ \forall i,j,k,m : (1 \leq i,k \leq n) (1 \leq j \leq \psi_i) (1 \leq m \leq \psi_k) ]
\]
\[
\implies t(H^x_j) \subseteq \{ H^x_m \} .
\]

We will only be concerned with clause sets that have an HL-proper lock numbering.

2.4.17 **HL-Sets**.

\[(\mathcal{H} \text{ an HL-set for } \mathcal{N})\]

iff
\[
[ (\text{unconstrained } \mathcal{H}) (\text{HL-proper } \mathcal{H}) (\mathcal{N} = 1(\mathcal{L}(\mathcal{H}))) ] .
\]

From this and previous definitions we have the following assertion:

\[(\mathcal{H} \text{ an HL-set for } \mathcal{N}^*)\]

\[
\implies [ (\text{unsatisfiable } \mathcal{H}) \text{ iff } (\text{unsatisfiable } \mathcal{N}^*) ] .
\]
2.4.18 Binary HLR.

Suppose that \( L \) a language of \( \{ \text{H}^\text{f}, \text{H}^\text{t} \} \) and that
\( \Psi \) a semantic function for \( L \). Then we define a 6 place relation (on \( \Psi \), 3 clauses, and 2 literals) of immediate
deducibility by a binary HLR inference rule application, as
follows:

\[
\begin{array}{c}
\text{H}^\text{f} \rightarrow \text{H}^\text{t} \\
\Psi, S^\text{f}, S^\text{t}
\end{array}
\]

iff

\[ \exists \sigma : \]

\[
\begin{align*}
& ( \text{t.v.d.}(\text{H}^\text{f}, \Psi) = \text{F} ) \lor ( \text{t.v.d.}(\text{H}^\text{t}, \Psi) = \text{T}/\text{F} ) \quad \text{c1} \\
& ( \text{t.v.d.}(\text{H}^\text{f}, \Psi) = \text{T} ) \lor ( \text{t.v.d.}(\text{H}^\text{t}, \Psi) = \text{T}/\text{F} ) \quad \text{c2} \\
& (S^\text{f} \in S(\text{H}^\text{f})) \quad \text{c3} \\
& (\forall H : H \in S(\text{H}^\text{f}) \rightarrow f(S^\text{f}) \leq f(H)) \quad \text{c4} \\
& (S^\text{t} \in S(\text{H}^\text{t})) \quad \text{c5} \\
& \text{(unifiable } \{S^\text{f}, \neg S^\text{t} \} \text{)} \quad \text{(c6)} \quad \sigma = \text{mgu}(\{S^\text{f}, \neg S^\text{t} \}) \\
& (S(\text{H}^\lambda) = ( (S(\text{H}^\text{f})) \sigma - \{S^\text{t}\} \sigma ) \cup (S(\text{H}^\text{t}) \sigma - (S^\text{f}) \sigma ) ) \quad \text{c7} \\
& (\forall(\text{H}^\lambda) = (\forall(\text{H}^\text{f}) \cup \forall(\text{H}^\text{t})) \sigma ) \quad \text{c8} \\
& \text{(} \Psi \text{-feasible } \text{H}^\lambda \text{)} \quad \text{c9}
\end{align*}
\]

\( S^\text{f} \) is called the false selected literal, and \( S^\text{t} \) the true
selected literal. \( \text{H}^\text{f} \) is called the false parent. \( \text{H}^\text{t} \) is
called the true parent, and in such an inference step \( \text{H}^\text{f} \) is
said to be resolved against \( \text{H}^\text{t} \) to produce the (binary)
resolvent \( \text{H}^\lambda \).
As an example of this definition we consider, from Figure 2--2, clauses 18, 9, and 24, and the inference step

\[
\begin{align*}
18 \times 9 & \vdash \text{BINARY} \quad \text{ANC} \\
\forall x \forall y \exists z & (D(a, x, b) \land D(a, y, z)) \\
\forall x & (D(a, x, y) \rightarrow D(a, y, z)) \\
24 & \end{align*}
\]

where \( \forall \) is the semantic function based on set theory which was used in Example II, and

clause 9 = \( \neg P(a)^{\text{coff}}, \neg M(b, z, u)^{\text{coff}}, \neg D(a, u)^{\text{coff}}, D(a, z)^{\text{coff}} \);

\( \text{FSL} = \{ \neg D(a, b), \neg P(a), \neg M(b, z, u), \neg D(a, u) \} \) T/F

clause 18 = \( \neg D(a, S(b))^{\text{coff}} \);

\( \text{FSL} = \{ \neg D(a, b), \neg P(a), \neg M(b, b, S(b)), \neg D(a; S(b)) \} \) F.

Conditions c1) and c2) of definition 2.4.18 are satisfied by taking clause 18 as the false parent and clause 9 as the true parent. Thus, in terms of the symbols used in definition 2.4.18,

\[ H^2 \text{ is clause 18} \]

\[ H^2 \text{ is clause 9} \]

and taking

\[ S_f \text{ as } \neg D(a; S(b))^{\text{coff}} \]

and

\[ S_t \text{ as } D(a, z)^{\text{coff}} \]

satisfies conditions c3), c4) and c5).

Condition c6) is satisfied by

\[ \sigma = \{S(b)/z\} \]

and c7), c8) and c9) are then satisfied by letting \( H^A \) be clause 24.
clause 24 = \neg P(a) \land \neg M(b, S(b), u) \land \neg D(a, u) \\
FSL = \{ \neg D(a, b), \neg P(a), \neg M(b, b, S(b)), \neg D(a, S(b)) \\
\neg M(b, S(b), u), \neg D(a, u) \} ^c

For the examples we write the FSL's as sets of literals without the lock numbers. Notice that the binary HLR inference rule is distinct from the rule used in Example 11 (Figure 2-2), even though for the particular case of clause 18 resolved against clause 9, the resolvent is the same.

Let \( \mathcal{S} \) be \( (H^k \times H^k | \psi, \sigma) \). \( \mathcal{S} \) is a relation (of immediate deducibility) in our meta-language, but we will take the informal notational liberty of saying that \( \mathcal{S} \) also is an immediate deduction of \( H^k \) from \( H^k \) against \( \neg \{ \} \) according to Binary HLR using \( \psi \) (and usually shorten this to " \( \mathcal{S} \) is a Binary HLR inference step"). We do the same thing also for the relations defined in definitions 2.4.19 through 2.4.22.

2.4.19 False Parent Factoring
Suppose \( \psi \) a language of \( H^k \) and \( \psi \) a semantic function for \( \psi \), then we define a 5-place relation (on \( \psi \), 2 clauses, a literal and a substitution) of one clause being a false factor of another, as follows:
(\mathcal{S}_f, \Psi, \text{false factor of } \mathcal{H}^z) \\
\text{iff} \\
\{ (t.v.d.(\mathcal{H}^z, \Psi) = F) \lor (t.v.d.(\mathcal{H}^z, \Psi) = T/F) \} \\
(\mathcal{S}_f \in \mathcal{S}(\mathcal{H}^z)) \\
(\forall H : H \in \mathcal{S}(\mathcal{H}^z) \implies f(\mathcal{S}_f) \leq f(H)) \\
(\exists U : (U \in \mathcal{S}(\mathcal{H}^z)) (\mathcal{S}_f \in U) \\
\text{(unifiable } U) \ (\exists = mg(u(U)) \\
(\mathcal{H}_f = < (\mathcal{S}_f) \cup \mathcal{S}(\mathcal{H}^z) \cup ^\mathcal{U} \cup (\mathcal{H}^z) \cup \mathcal{S}(\mathcal{H}^z) \cup ^\mathcal{U} \cup >) \}) \\
(\Psi, \text{feasible } \mathcal{H}_f) \}

\mathcal{S}_f \text{ is called the selected literal for factoring in } \mathcal{H}^z.

As an example of definition 2.4.19, consider first

\mathcal{H}^z = \mathcal{Q}1:

\mathcal{Q}1 = \{ \leq(x,y)_{1001}, \leq(y,x)_{1002}, =, (x,y)_{333}^3 ; \ FSL = () \}

with the usual interpretation of "\leq" and "\" for real numbers determining \(\Psi\). Then there is no \(\Psi\)-false factor of \mathcal{Q}1 since t.v.d.(\mathcal{Q}1, \Psi) = T.

Using the same \(\Psi\), if we instead take \mathcal{H}^z = \mathcal{Q}2:

\mathcal{Q}2 = \{ \leq(x,y)_{1001}, \leq(y,x)_{1002}, =, (x,y)_{333}^3 ; \ FSL = () \}

then (1) is satisfied, and if we take

\mathcal{S}_f = \{ \leq(x,y)_{1001} \\
\mathcal{U} = \{ \mathcal{S}_f \} \\
\mathcal{V} = () \}

then we have the \(\Psi\)-false factor.
Another \( \psi \)-false factor of \( Q \) is obtained by taking

\[ Q'_f = \sim \leq(x, z)_{\text{root}}, \sim \leq(z, z)_{\text{root}}; \]

\[ FSL = \{ \sim \leq(z, z), \sim \leq(z, z) \} \quad F. \]

Notice that definition 2.4.19 does not require that factoring be maximal in any sense, so that if the standard literals of a clause are

\[ P(x)'_{\text{root}}, P(a)'_{\text{root}}, P(z)'_{\text{root}}; \]

then, assuming condition c7 is satisfied, it is intended that one of the factors of the clause will have the standard literal set

\[ P(a)'_{\text{root}}, P(a)'_{\text{root}}; \]

(obtained by letting \( S_f = P(x)'_{\text{root}} \) and \( U = \{ S_f, P(a)'_{\text{root}} \} \)).

2.4.20 True Parent Factoring.

Suppose (\( L \) a language for \( \Sigma \)) and (\( \psi \) a semantic function for \( L \)), then we define a 5 place relation, indicating one clause is a true factor of another, for true factoring analogous to that for false factoring as follows:
($S_t \in \mathcal{S}({\mathcal{H}}^\psi)$)

iff

$\exists \Omega :$

\[
(\text{t.v.d.}({\mathcal{H}}^\psi, \psi) = \text{T}) \lor \ (\text{t.v.d.}({\mathcal{H}}^\psi, \psi) = \text{T/F})
\]

($S_t \in \mathcal{S}({\mathcal{H}}^\psi)$)

($\Omega = \{ H \mid H \in \mathcal{S}({\mathcal{H}}^\psi) \land t(H) < t(S_t) \}$)

($\exists U :$ (U $\subseteq \mathcal{S}({\mathcal{H}}^\psi)$) ($S_t \in U$)

(unifiable U) ($\xi = \text{mgu}(U)$)

(\mathcal{H}_t = \{(S_t, \xi)\} \cup (\mathcal{S}({\mathcal{H}}^\psi) - U)\xi)

$\exists \xi \in \mathcal{S}({\mathcal{H}}^\psi)$

($\psi$-feasible $\mathcal{H}_t$)

$S_t$ is called the selected literal for factoring in $\mathcal{H}^\psi$.

As an example of definition 2.4.20, let $\mathcal{H}^\psi$ be clause 6 of Example II:

clause 6 = $\neg P(x)^v_{001}$, $\neg M(y, z, u)^v_{001}$, $\neg D(x, u)^v_{010}$, $D(x, y)^v_{011}$, $D(x, z)^v_{012}$;

FSL = $\{\}$ T

and let $\psi$ be determined by the model used in Example II.

By taking

$S_t = D(x, y)^v_{011}$

$\Omega = \{\neg P(x)^v_{001}, \neg M(y, z, u)^v_{001}, \neg D(x, u)^v_{010}\}$

$U = \{D(x, y)^v_{011}, D(x, z)^v_{012}\}$

$\xi = \{w/z, q/y, q/z\}$

we get the $\psi$-true factor of clause 6

$D(w, q)^v_{011}$, $\neg P(w)^v_{001}$, $\neg M(q, q, u)^v_{001}$, $\neg D(w, u)^v_{001}$;

FSL = $\{\neg D(w, q), \neg P(w), \neg M(q, q, u), \neg D(w, u)\}$ T.
If instead we take $S_\epsilon$ and $\Omega$ as above, but let
\[ U = \{D(x,y)_{\text{not}}'\} \]
\[ \epsilon = \{\} \]
then we get the factor
\[ D(x,y)_{\text{not}}'', P(x)^p, M(y,z,u)_{\text{not}}, D(z,v)_{\text{not}}, D(x,z)_{\text{not}}; \]
\[ \text{FSL} = \{D(x,y), P(x), M(y,z,u), D(x,u)\} T. \]

Yet another possible $\Psi$-true factor of clause 6 is
\[ D(x,z)_{\text{not}}'', P(x)^p, M(y,z,u)_{\text{not}}, D(x,u)_{\text{not}}, D(x,y)''; \]
\[ \text{FSL} = \{D(x,z), P(x), M(y,z,u), D(x,u), D(x,y)\} T. \]

This exhausts the true factoring possibilities on the literal letter "D". The other literals also can be the selected literals for factoring. For example, we could take
\[ S_\epsilon = P(x)^p \]
\[ \Omega = \{\} \]
\[ U = \{P(x)^p\} \]
\[ \epsilon = \{\} \]
which gives the factor
\[ P(x)^p, M(y,z,u)_{\text{not}}, D(x,u)_{\text{not}}, D(x,y)''; \]
\[ \text{FSL} = \{P(x)\} T \]
and this is the only true factoring possibility with $P(x)^p$ as the selected literal in clause 6. Two other true factors are possible, one each with selected literal $M(y,z,u)_{\text{not}}$ and selected literal $D(x,u)_{\text{not}}$.

As in the case of $\Psi$-false factoring, the $\Psi$-true factoring is not required to be maximal in any sense.
2.4.21 The HLR Inference Rule.
Suppose \( (L, (\mathcal{H}^x, \mathcal{H}^z)) \) and \( (\psi; \tau) \), then we define the 4 place relation (on \( \psi \) and 3 clauses) indicating that a clause is derivable from two other clauses by a single application of the HLR inference rule, as follows:

\[
(\mathcal{H}^x \times \mathcal{H}^z \mid \psi \tau \rightarrow \mathcal{H}^k)
\]

iff

\[
\exists \xi_1 \exists \xi_2 \exists \xi_3 \exists \xi_4 \exists H_k \exists H_L : \\
(\xi_1)_{\mathcal{H}^x} \psi \text{ false factor of } \mathcal{H}^x \hspace{1cm} c_1 \\
(\xi_2)_{\mathcal{H}^z} \psi \text{ true factor of } \mathcal{H}^z \hspace{1cm} c_2 \\
(\mathcal{H}^x \times \mathcal{H}^z \mid \psi \tau \rightarrow \mathcal{H}^k) \hspace{1cm} c_3
\]

\( S_f \) and \( S_b \) are the selected literals of \( \mathcal{H}^x \) and \( \mathcal{H}^z \) respectively. Note that \( \mathcal{H}^x \) and \( \mathcal{H}^z \) may be the same clause.
When this is the case we say that the inference step is an occurrence of self resolution.

As an example of definition 2.4.21 we let \( \mathcal{H}^x \) be clause 7 and \( \mathcal{H}^z \) be clause 6, of Example II (Figure 2--2).
Clauses 9 and 10 of Figure 2--2 are two examples of HLR resolvents of clause 7 against clause 6. In Example II no factoring was done which actually "factored away" any standard literals. We show here a clause, \( \mathcal{H}^k \), immediately deducible from clause 7 and clause 6, but which is not in the search space at level 1 of Example II. We have:

\[ \mathcal{H}^x = \text{clause 7 } = \text{"D}(a, b)^{1/3} ; \text{ FSL = } \{ \} \text{ F} \]
\[ H^* = \text{clause 6} = \]
\[ \neg P(x), \neg H(y, z, u), \neg D(x, u), \neg D(x, y), \neg D(x, z); \]
\[ \text{FSL} = \{ \} \]

\[ H' = \neg D(a, b); \quad \text{FSL} = \{ \neg D(a, b) \} \]

\[ H'' = D(w, q), \neg P(w), \neg M(q, q, u), \neg D(w, u); \]
\[ \text{FSL} = \{ \neg D(w, q), \neg P(w), \neg M(q, q, u), \neg D(w, u) \} \]

where \( \psi \) is again determined by the model used in Example II, and where

\[ S_\xi = \neg D(a, b) \]
\[ \emptyset = \{ \} \]
\[ S_\omega = D(x, y) \]
\[ \xi = \{ w/x, q/y, q/z \} \]

Conditions c1 and c2 are satisfied by these choices, and c3 is satisfied by taking \( H = Q \):

\[ Q = \neg P(x), \neg H(y, z, u), \neg D(x, u); \]
\[ \text{FSL} = \{ \neg D(a, b), \neg P(a), \neg M(a, b, u), \neg D(a, u) \} \]

Thus we have

\[ \text{clause 7} \times \text{clause 6} \quad \vdash \frac{\psi}{\neg \psi} \quad \text{Q} \]

as well as

\[ \text{clause 7} \times \text{clause 6} \quad \vdash \frac{\psi}{\neg \psi} \quad \text{clause 9} \]

\[ \text{clause 7} \times \text{clause 6} \quad \vdash \frac{\psi}{\neg \psi} \quad \text{clause 10}. \]

Notice that clause 12 of Figure 2--2 is equal to Q, but clause 12 is not deduced from clauses 7 and 6 in one inference step because factoring was not used in Example II.
2.4.22 HL-Deductions and Refutations.

Let \( \mathcal{J} \in (\mathcal{E} | -\frac{\text{HL}}{\psi} \rightarrow \mathcal{H}) \).

If \( \mathcal{J} \) then we say that clause \( \mathcal{H} \) is deducible from the set of clauses \( \mathcal{E} \) by the HLR inference rule, using \( \psi \).

\( \mathcal{J} \) iff there exists a finite, strict binary, rooted tree, \( \mathcal{T} \), such that:

1. each leaf is a \( \psi \)-feasible clause in \( \mathcal{E} \);
2. for each internal node, \( \mathcal{H} \), with parents \( \mathcal{H}^\prime \) and \( \mathcal{H}^\ast \), it is the case that either
   \[
   \mathcal{H}^\prime \times \mathcal{H} \times \mathcal{H}^\ast | -\frac{\text{HL}}{\psi} \rightarrow \mathcal{H}^\mathcal{H}
   \]
   or
   \[
   \mathcal{H}^\prime \times \mathcal{H} \times \mathcal{H}^\ast | -\frac{\text{HL}}{\psi} \rightarrow \mathcal{H}^\mathcal{H}.
   \]
3. \( \mathcal{H} \) is the root.

Such a tree is called an HL-deduction (or HLR deduction) of \( \mathcal{H} \) from \( \mathcal{E} \) using \( \psi \). If \( \mathcal{H} \) is a null clause, then the tree is called an HL-refutation of \( \mathcal{E} \) (using \( \psi \)). If \( \mathcal{T} \) is a one node tree or a three node tree, then \( \mathcal{H} \) is said to be immediately deducible from \( \mathcal{E} \).

2.4.23. The main discussion of HLR appears in Chapter 5, but some detailed comments are appropriate here for the preceding definitions. Definition 2.4.18 (Binary Resolvent) is rather weak with respect to both the syntactic condition (no restriction is placed on the literal \( \mathcal{C}_b \) with respect to its lock number) and its semantic condition (the FSL of the resolvent is just the union of the parent FSL's). The HLR inference rule, definition 2.4.21, however, does not suffer
from these weaknesses because it requires that the binary resolution step (condition c3 of definition 2.4.21) occur on specific literals in \( \mathcal{V} \)-false and \( \mathcal{V} \)-true factors. As can be seen from the factoring definitions (2.4.19 and 2.4.20) much of the restrictiveness of the NLR inference rule has been built into the factoring operation. Also notice that definition 2.4.21 does not have an explicit feasibility condition, as it is there indirectly in 2.4.18.

It is now possible to state the relationship between the NLR inference rule of definition 2.4.21 and the inference rule used in Examples I and II. In the examples no factoring was done in the sense of factoring away any standard literals. However the inference rule used in the examples formed resolvents whose FSL sets contained more literals than the union of the FSL sets of their parents (in distinction to definition 2.4.18). The reader should be able to see that the inference rule used in the examples was precisely that given in definition 2.4.21 with the restriction that the \( \mathcal{V} \)-false and \( \mathcal{V} \)-true factors used in 2.4.21

1. use empty substitutions (\( \mathcal{V} \) and \( \mathcal{Q} \) of 2.4.21);

ii. not factor away any literals (i.e. \( U = \{ S_\mathcal{Y} \} \) in 2.4.19, and \( U = \{ S_\mathcal{X} \} \) in 2.4.20).

Since NLR is a refinement of unrestricted resolution, such a restriction on the factoring gives a refutation incomplete inference system, although a refutation still does exist for the clause set of Example II with this restricted factoring.
2.5 Soundness of HL-Resolution.

The soundness of HLR is easy to prove, since it is a refinement of unrestricted resolution. In HLR, clauses have t.v.d. values, but do not have the usual truth values relative to a model defined for them (with the exception of unconstrained clauses, as in definition 2.4.8.5). Thus we state a deduction soundness theorem in terms of the sets of literal cores of the standard literals of the clauses involved in HL-deductions.

Theorem 2.5 — 1

(Deduction Soundness of HL-Resolution)

\[ \exists \sigma : \quad ( \lambda \text{ a language for } \mathcal{H}) \]

\[ (\sigma \text{ a semantic function for } \lambda) \]

\[ (\mathcal{H} \vdash \sigma(\mathcal{H}) \rightarrow \sigma(\lambda)) \]

\[ \Rightarrow \]

\[ \lambda(\sigma(\mathcal{H})) \models \lambda(\sigma(\lambda)) \].

proof: Let $\Gamma$ be any HL-deduction tree of $\mathcal{H}$ from $\mathcal{H}$, using $\sigma$, that exists by virtue of $\mathcal{H} \vdash \sigma(\mathcal{H}) \rightarrow \mathcal{H}$. Replace each node $\mathcal{H}^\epsilon$, in $\Gamma$, by the normal clause $\mathcal{N}^\epsilon$, where

\[ \mathcal{N}^\epsilon = \lambda(\sigma(\mathcal{H}^\epsilon)) \]

to form the tree $\Gamma$. By examining definitions 2.4.19 through 2.4.21 it should be clear that $\Gamma$ is a deduction of $\lambda(\sigma(\mathcal{H}))$ from $\lambda(\sigma(\mathcal{H}))$ according to the inference rule of unrestricted resolution. But unrestricted resolution is sound (e.g. see [Nilsson, 1971]) and thus c4 is true. Q.E.D.
We next state and prove two corollaries giving the refutation soundness of HLR.

Corollary 2.5 -- 2
(Refutation Soundness)

\[ \exists \psi : ( \psi \text{ a language for } \mathcal{L} ) \]
(\psi \text{ a semantic function for } L)
(unconstrained \( \mathcal{L} \))
(null \[ \exists \])
(\( \mathcal{L} \models \frac{\mathcal{L}}{\psi} \))

\[ \Rightarrow \]
(unsatisfiable \( \mathcal{L} \)).

proof: We assume c1 - c5 are true. Then by Theorem 2.5-1,

p1. \( \land(\mathcal{L}(\mathcal{L})) \models [ ] \).

p2. From definition 2.4.8.2

\[ \forall h : ( h \text{ a Herbrand interpretation for } L ) \]
\[ \Rightarrow \phi_\Lambda([ ]) = \text{false}. \]

p3. Then from definition 2.4.8.7 and p1 and p2,

\[ \forall h : ( h \text{ a Herbrand interpretation for } L ) \]
\[ \Rightarrow \phi_\Lambda(\land(\mathcal{L}(\mathcal{L}))) = \text{false} \].

p4. Finally from p3 and definitions 2.4.8.5 and 2.4.11 we have (unsatisfiable \( \mathcal{L} \)).

Q.E.D.
Notice that in the above Corollary c3 is necessary, since unsatisfiability has not been defined for constrained clauses (see definition 2.4.11).

The next Corollary is the primary soundness result of interest in practical theorem proving, as it gives the overall soundness of the process of taking a set of normal clauses, forming an HL-set for them, and asserting the unsatisfiability of the normal set on the basis of the existence of an HL-refutation of the HL-set.

Corollary 2.5 -- 3
(Refutation Soundness)

I Û: ( L a language for N )
( Û a semantic function for L )
( N an HL-set for N )
(null N )
( N | - Û ϕ (E ) )

----> (unsatisfiable N ).

proof: Assume c1 - c5 are true.

p1. Then by c3 and definition 2.4.17,

(unconstrained N ).

p2. From c4, c5, p1 and Corollary 2.5-2,

(unsatisfiable N ).

p3. Let h be an arbitrary Herbrand interpretation for L.
Then by p1, p2, and definition 2.4.11,

$$\phi_4(N') = \text{false},$$

and thus $$\phi_4(1(S(N'))) = \text{false},$$ by definition 2.4.8.5. But then by c3 and definition 2.4.17, we have $$\phi_4(N') = \text{false},$$ and thus (unsatisfiable $$N'$$), by definition 2.4.11.

Q.E.D.

Notice that none of the soundness results require that the semantic function be sound. In the next chapter the completeness of MLR will be dealt with, and there it is necessary for the semantic functions to be sound.
Chapter 3

COMPLETENESS OF HL-RESOLUTION

In this chapter the refutation completeness of HLR is established under fairly general conditions. The primary theorem is Theorem 3.1-1 which asserts the refutation completeness of HLR using any sound semantic function. Theorem 3.1-2 is a special case of this, and asserts the refutation completeness of HLR using the semantic function $\phi_{\mathcal{A}}$ for any arbitrary Herbrand interpretation for any language of the clauses. Theorem 3.1-2 is a convenient intermediate point in the overall proof of 3.1-1, as well as being of some interest in itself.

3.1 Completeness Theorems.

Theorem 3.1-1

(False Permissive Completeness of HLR)

\[
\begin{array}{l}
\text{(HL-proper $\mathcal{H}$)} \\
\text{(unconstrained $\mathcal{H}$)} \\
\text{(unsatisfiable $\mathcal{H}$)} \\
\text{(L a language for $\mathcal{H}$)} \\
\text{($\mathcal{V}$ a sound semantic function for L)} \\
\end{array}
\]

\[
\implies
\exists \mathcal{A} \exists \mathcal{X} \exists \mathcal{Y} \exists \mathcal{G} \exists \mathcal{E}
\]

\[
\text{(null $\mathcal{X}$)}
\]

\[
\text{($\mathcal{H} \vdash_{\mathcal{V}} \mathcal{G} \mathcal{E}$ $\mathcal{A}$)}
\]

\[
c1 \quad c2 \quad c3 \quad c4 \quad c5 \]
Theorem 3.1-1 is the theorem of main interest since, as will be shown in Chapter 4, the use of a semantic function which is required only to be sound allows the use of models which are quite general in character yet are computationally tractable.

Theorem 3.1-2
(Herbrand Model Completeness of HLR)

\[ \text{(HL-proper } \mathcal{H} \text{)} \]
\[ \text{(unconstrained } \mathcal{H} \text{)} \]
\[ \text{(unsatisfiable } \mathcal{H} \text{)} \]
\[ \text{(L a language for } \mathcal{H} \text{)} \]
\[ \text{(h a Herbrand interpretation for L)} \]
\[ \text{----->} \]
\[ \begin{align*}
\exists \mathcal{F} &\exists \mathcal{G} \ (\text{null } \mathcal{F}) \\
\mathcal{H} &\vdash \phi\mathcal{H} \quad (\mathcal{H} \vdash \phi\mathcal{H})
\end{align*} \]

Theorem 3.1-2 gives the refutation completeness of HLR using a semantic function which is the semantic function for any Herbrand interpretation, h (for an appropriate language).

Theorem 3.1-1 states in effect that for any h, \( \phi\mathcal{H} \) may be replaced by any semantic function \( \psi \) which is false permissive with respect to \( \phi\mathcal{H} \), and we will still have refutation completeness of HLR. Because of this property we
say that HLR is false permissive complete. In general if a refinement of resolution is a refinement that depends upon a semantic function, and it is refutation complete when any sound semantic function is used, then we say that the refinement is false permissive complete.

From Theorem 3.1-1 we have immediately the false permissive completeness of TMS and TMS + FSL, since HLR is a refinement of both of these. It is easy to show the false permissive completeness of semantic P-deduction (Slagle, 1967), semantic P-deduction with FSL's, and positive-unit semantic resolution for Horn sets (Henschel, 1975) with and without FSL's. The nature of the completeness proofs for these other refinements should be obvious to the reader from the proofs of theorems in this chapter.

From definitions 2.4.8, 2.4.11, and 2.4.17 we have

\[ \{ (\not\vdash \text{an HL-set for } \not\exists) \rightarrow (\text{unsatisfiable } \not\exists) \} \]

Thus refutation completeness results for unsatisfiable sets, \( \not\vdash \), translate immediately into refutation completeness results for the overall procedure of taking a set of normal clauses, \( \not\vdash \), forming any HL-set for \( \not\vdash \), and applying the HLR inference rule to it.
3.2 Herbrand Model Completeness Proof.

The proof of Theorem 3.1-2 follows a common argument in resolution completeness proofs in that HLR using $\Phi^H$ will be shown to be complete at the ground level first, and then the general level result is obtained by the use of a lifting lemma.

3.2.1 $H$-augmentations for Ground Clauses.

We define the following 3-place relation on two clauses and one literal:

\[
\{\cdot\}^* \text{ an } H\text{-augmentation of } \{\cdot\}
\]

iff

\[
\{H\} = < \mathcal{L}(\{\cdot\}^*) - K_\mathcal{F}, \mathcal{F}(\{\cdot\}^*) - K_{\mathcal{F}} >
\]

where $K_\mathcal{F}$ and $K_{\mathcal{F}}$ are each (independently) either the empty set or the singleton set \{H\}.

We require the notion of an $H$-augmentation only for ground clauses.
Lemma 3.2

\[ (\models \frac{x}{x} \text{ an } \mathcal{H}\text{-augmentation of } \mathcal{H}^\sharp ) \]

\[ (\models \frac{x}{x} \text{ an } \mathcal{H}\text{-augmentation of } \mathcal{H}^\ast ) \]

\[ ( \text{ground } \models \frac{x}{x} ) \]

\[ ( \text{ground } \models \frac{x}{x} ) \]

\[ ( L \text{ a language for } \{ \mathcal{H}^\sharp, \mathcal{H}^\ast \} ) \]

\[ ( h \text{ a Herbrand interpretation for } L ) \]

\[ ( \text{ground } H ) \ ( \phi_h(H) = \text{false} ) \]

\[ ( \forall \hat{h} : \left( \left( \hat{h} \in \left( \mathcal{S}(\frac{x}{x}) \cup \mathcal{S}(\frac{x}{x}) \right) \right) \ ( \phi_h(\hat{h}) = \text{false} \) \right) \]

\[ \rightarrow f(\hat{h}) \leq f(H) \)

\[ ( \mathcal{H}^\sharp \times \mathcal{H}^\ast \mid \frac{\phi_h}{\phi_h} \models \mathcal{H}^\sharp ) \]

\[ \rightarrow \]

\[ \exists \mathcal{H}^\Delta \mid ( \mathcal{H}^\Delta \text{ an } \mathcal{H}\text{-augmentation of } \mathcal{H}^\Delta ) \]

\[ ( \mathcal{H}^\sharp \times \mathcal{H}^\ast \mid \frac{\phi_h}{\phi_h} \models \mathcal{H}^\sharp ) \].

**proof:** Assume cl - c9 true. The proof is by cases. The trivial case is when \( \models \frac{x}{x} = \mathcal{H}^\sharp \) and \( \mathcal{H}^\sharp = \mathcal{H}^\ast \), for then taking \( \mathcal{H}^\Delta = \mathcal{H}^\Delta \) will satisfy c10 and c11. The two cases to be argued explicitly below are:

**Case 1.** \( \mathcal{H}^\sharp \models \mathcal{H}^\ast \) and \( \mathcal{H}^\ast \not\subseteq \mathcal{H}^\sharp \), and either \( \mathcal{S}(\frac{x}{x}) = \mathcal{S}(\frac{x}{x}) \) or \( \mathcal{F}(\frac{x}{x}) = \mathcal{F}(\frac{x}{x}) \).

**Case 2.** Identical to case 1 with i and j interchanged.

All remaining cases can then be covered by an iterative application of the Lemma for these two cases, using the fact that "an \( \mathcal{H}\text{-augmentation of}" is a transitive relation. Note that even for a given \( \mathcal{H}^\sharp \) and \( \mathcal{H}^\ast \), there will not
necessarily be only one \( L^k \) which satisfies \( c9 \). Thus the cases below argue the existence of an \( L^k \) for the lemma consequent such that \( L^k \) is to be constructed from \( L^k \).

Case 1. \( L^k = L^k \) and \( L^k \preceq L^k \), and

either \( \mathcal{S}(L^k) = \mathcal{S}(L^k) \) or \( \mathcal{F}(L^k) = \mathcal{F}(L^k) \).

Case 1.1 \( \mathcal{S}(L^k) = \mathcal{S}(L^k) \).

The \( L^k \) that the lemma consequent asserts exists will be

\( L^k = \langle \mathcal{S}(L^k), (\mathcal{F}(L^k) \cup \{H\}) \rangle \).

The immediate deduction

\[ L^k \times L^k \vdash L^k \]

exists if \( c1 - c9 \) are all true, since all the necessary conditions in definitions 2.4.18 through 2.4.21 can be satisfied. In particular \( L^k \) is feasible and \( L^k \) is feasible because of \( c7 \). Likewise because of \( c7 \) the t.v.d. values of the starred clauses is the same as the corresponding unstarred clauses.

Case 1.2 \( \mathcal{F}(L^k) = \mathcal{F}(L^k) \).

Since \( L^k \) and \( L^k \) are ground and feasible, we have from \( c7 \) and \( c9 \),

\[ t.v.d.(L^k, \phi_k) = t.v.d.(L^k, \phi_k) = F. \]

It is possible to satisfy all of the conditions in definitions 2.4.18 through 2.4.21 and generate an \( L^k \) such that

i. \( \mathcal{S}(L^k) = \mathcal{S}(L^k) \cup \{H\} \)

(if the selected literal in \( L^k \) is unifiable with \( H \), then one may take, optionally, \( \mathcal{S}(L^k) = \mathcal{S}(L^k) \))

ii. \( \mathcal{F}(L^k) = \mathcal{F}(L^k) \cup \{H\} \).
Such an $\mathcal{H}^*$ will satisfy c10 and c11.

Case 2. $\mathcal{H}^* = \mathcal{H}$ and $\mathcal{H}^* \not\subseteq \mathcal{H}$ and

either $\mathcal{S}(\mathcal{H}^*) = \mathcal{S}(\mathcal{H})$ or $\varphi(\mathcal{H}^*) = \varphi(\mathcal{H})$.

Case 2.1 $\mathcal{S}(\mathcal{H}^*) = \mathcal{S}(\mathcal{H})$.

Essentially the same as case 1.1.

Case 2.2 $\varphi(\mathcal{H}^*) = \varphi(\mathcal{H})$.

It is easy to verify that

$$t.v.d.(\mathcal{H}^*, \phi_\lambda) = t.v.d.(\mathcal{H}, \phi_\lambda) = T$$

because of c4, c7 and c9.

The set $\Omega$ in definition 2.4.20.c3 may contain an extra occurrence of $\mathcal{H}$ in the deduction all of this lemma, compared to c9, depending upon the true lock number of $\mathcal{H}$.

All of the conditions in definitions 2.4.18 through 2.4.21 can be satisfied, producing a $\mathcal{H}^*$ such that

i. $\mathcal{S}(\mathcal{H}^*) = \mathcal{S}(\mathcal{H}) \cup \{H\}$

ii. $\varphi(\mathcal{H}^*) = \varphi(\mathcal{H}) \cup \{H\}$ if $H$ is in $\Omega$ of definition 2.4.20.c3, otherwise $\varphi(\mathcal{H}^*) = \varphi(\mathcal{H})$.

Q.E.D.

3.2.7 Excess Literal Parameter. Let $E$ be a function mapping clauses and clause sets to their excess literal parameter (Anderson and Bledsoe, 1970), defined by:

$$E(\mathcal{H}) = \text{one less than the number of literals in } \mathcal{S}(\mathcal{H})$$

$$E(\mathcal{N}) = \sum_{\mathcal{H} \in \mathcal{N}} E(\mathcal{H})$$
Theorem 3.2 -- 2

(Ground Herbrand Model Completeness -- Extended)

\(
\{ \\
\text{(ground } \mathcal{H} \text{)} \\
\text{(HL-proper } \mathcal{H} \text{)} \\
\text{(L a language for } \mathcal{H} \text{)} \\
\text{(h a Herbrand interpretation for } L \text{)} \\
\text{(} \forall \mathcal{H} : \mathcal{H} \in \mathcal{H} \implies (\text{ } \mathcal{H} \text{-feasible } \mathcal{H}) \text{)} \\
\text{(unsatisfiable } \lambda(\mathcal{L}(\mathcal{H})) \text{)} \\
\} \\
\implies \\
\exists \mathcal{H} : \{ \text{(null } \mathcal{H}) \\
\text{(} \mathcal{H} \vdash \frac{\mathcal{L}(\mathcal{H})}{\mathcal{H}} \text{)} \}.
\)

The reason this theorem is called "extended" is that, in distinction to Theorem 3.1-2, \( \mathcal{H} \) is not required to be unconstrained. Notice that \( c5 \) and \( c6 \) of this theorem are implied by

\(
\{ \\
\text{(unconstrained } \mathcal{H}) \\
\text{(unsatisfiable } \mathcal{H}) \}
\)

so that the non-extended form of this theorem (i.e. \( c5 \) and \( c6 \) replaced by \( c5' \) and \( c6' \)) is a special case of the extended form. The non-extended form is the ground analogue of Theorem 3.1-2. By the Compactness theorem (and various definitions in section 2.4) it is sufficient to argue the proof of Theorem 3.2-2 under the assumption that \( \mathcal{H} \) is a finite set.
proof: The proof is by induction on the excess literal parameter of \( \mathcal{H} \), \( E(\mathcal{H}) \). Assume \( c_1 - c_6 \) all true.

basis: \( E(\mathcal{H}) \leq 0 \).

Case 1. There exists a null clause in \( \mathcal{H} \). Conditions \( c_7 \) and \( c_8 \) are then true.

Case 2. There is no null clause in \( \mathcal{H} \). Thus \( E(\mathcal{H}) = 0 \) and each clause in \( \mathcal{H} \) has exactly one standard literal. Then by \( c_6 \) there must exist two clauses in \( \mathcal{H} \) of the form

\[
\mathcal{H}' = \langle i_1, N, i_2 \rangle, \quad \mathcal{H}'' = \langle i_3, \neg N, i_4 \rangle
\]

where \( N \) is a normal ground literal false in \( h \), and \( i_1, i_2, i_3, i_4 \) are integers. By definitions 2.4.18 through 2.4.21 we have

\[
\mathcal{H}' \times \mathcal{H}'' \not\models \varphi(\mathcal{H}') \cup \varphi(\mathcal{H}'')
\]

and clearly \( c_7 \) and \( c_8 \) are true.

step: \( E(\mathcal{H}) = N > 0 \), and the theorem is true for all clause sets of excess literal parameter less than \( N \). If a clause in \( \mathcal{H} \) is a null clause then we have the step trivially. Otherwise there are two cases.
Case 1. There exists a literal \( h \) and a clause \( H \) meeting the following conditions:

\[
\begin{align*}
& (h \in \mathcal{H}) \quad \text{c9} \\
& (E(h) > 0) \quad \text{c10} \\
& (h \in \mathcal{S}(H)) \quad \text{c11} \\
& (\phi(h) = \text{false}) \quad \text{c12} \\
& (\forall h^* \exists h : [h^* \in \mathcal{S}(H^*) \land (H^* \in \mathcal{H})] \\
& \quad (\phi(h^*) = \text{false}) \land (f(h) < f(h^*)) \\
& \quad \implies (E(h^*) = 0) \quad \text{c13}
\end{align*}
\]

In this case the step proceeds in a fashion similar to the proof of the completeness of Hoyer's lock resolution (Chang and Lee, 1973).

Let \( \mathcal{H}' = (\mathcal{H} - \{h\}) \cup (\mathcal{S}(\{h\}) - \{h\}, \mathcal{F}(\{h\}) >) \). Now \( \mathcal{H}' \) can take the place of \( \mathcal{H} \) in the theorem statement, and conditions c1 - c6 are all true. Also \( E(\mathcal{H}') < N \), so by the induction step hypothesis there exists an HLR refutation, \( D_1 \), of \( \mathcal{H}' \), using \( \phi_A \). Modify \( D_1 \) to form \( D_1' \) by putting \( h \) back into the standard literal sets of all of the leaf occurrences of the clause

\[
\langle \mathcal{S}(\{h\}) - \{h\}, \mathcal{F}(\{h\}) >, \nonumber
\]

and applying Lemma 3.2-1 to successive levels of \( D_1 \), so that the root of \( D_1', R_1' \), will be such that

\[
(R_1' \text{ and } h\text{-augmentation of } R_1)
\]

where \( R_1 \) is the null clause at the root of \( D_1 \). (N.B. Lemma 3.2-1 is not necessarily applicable to every inference step in \( D_1 \), but the exceptions can also be transformed to yield
the same result, namely an H-augmentation of the original clause.)

Case 1.1. R1' is a null clause. The induction step is complete in this case.

Case 1.2. R1' is not a null clause. Then R1' has the form

\[ R1' = \langle \{H\}, \mathcal{F}(R1') \rangle \]

where \[ \mathcal{F}(R1') = \{ r_1, r_2, \ldots, r_n \} \]

for some integer \( n > 0 \).

Note that \( r_i = H \) or \( r_i \notin \mathcal{F}(R1) \), \( i = 1, 2, \ldots, n \). Thus each \( r_i \) is a false ground literal, making R1' feasible.

Consider the clause set

\[ \mathcal{K}_2 = (\mathcal{N} - \{H\}) \cup \{ \langle \{H\}, \mathcal{F}(R1') \rangle \} \]

\( \mathcal{K}_2 \) can take the place of \( \mathcal{N} \) in the theorem statement and conditions c1 - c6 are all true. Also \( \mathcal{E}(\mathcal{K}_2) < N \), so by the induction step hypothesis there exists an HLR refutation, D2, of \( \mathcal{K}_2 \), using \( \phi_{\mathcal{K}} \). Combining D1' and D2 will then give an HLR refutation of \( \mathcal{N} \), using \( \phi_{\mathcal{K}} \), and the induction step is complete in this case.

Case 2. There exists no \( H \) and \( \mathcal{N} \) satisfying conditions c9 - c13. But then, because of c6, it must be the case that there exists a subset of \( \mathcal{N} \), \( \mathcal{N}_F \), consisting of clauses each with only one standard literal, and that literal is false in \( h \), and a single clause, \( H_T \), consisting of standard literals all true in \( h \), such that

\[ 1(\mathcal{N}(\mathcal{N}_F)) \cup 1(\mathcal{N}(\{H_T\})) \]

is unsatisfiable. (N.B. If 1(\( \mathcal{N}(\{H_T\}) \)) is a singleton set,
then this case has in effect been already covered in the basis of the induction argument, so we assume this is not the case.)

Let \( \hat{H} \) be a standard literal occurrence of \( H_T \) of highest true lock number among the standard literals of \( H_T \). Form the clause set

\[
\mathcal{H}_3 = \{ < S(H_T) - \{\hat{H}\}, \varphi(H_T) > \} \cup \mathcal{H}_F.
\]

By the induction hypothesis, and the fact that \( E(\mathcal{H}_3) < N \), there will exist an HLR refutation, \( D_3 \), of \( \mathcal{H}_3 \), using \( \varphi_H \). Because of the structure of the clauses in \( \mathcal{H}_3 \), \( D_3 \) can only be an input deduction in which every internal node contains no false standard literals. Expressed differently, \( D_3 \) will be a linear refutation of \( \mathcal{H}_3 \), with top clause

\[
< S(H_T) - \{\hat{H}\}, \varphi(H_T) >
\]

and side clauses all in \( \mathcal{H}_F \), and with center clauses containing only true standard literals. Because \( D_3 \) has this special form, we can add one occurrence of \( \hat{H} \) as a standard literal to every clause in \( D_3 \) which is not in \( \mathcal{H}_F \), and the result is an HLR deduction, \( D_3' \), from \( \mathcal{H} \), using \( \varphi_H \), of a clause containing one standard literal, \( \hat{H} \). Let \( R3' \) be the root of \( D_3' \). There will exist a clause in \( \mathcal{H}_F \) whose only standard literal, \( \hat{H}* \), is such that \( l(\hat{H}*) = l(\hat{H}) \). Extending \( D_3' \) by an HLR inference step between this clause as false parent, and \( R3' \) as true parent gives an HLR refutation of \( \mathcal{H} \) using \( \varphi_H \) (in fact an input refutation), and completes the induction step in this case.

Q.E.D.
Lemma 3.2 -- 3

(MLR Lifting Lemma)

\[
\begin{array}{l}
\{ (L \text{ a language of } \{H', H^\omega\} ) \\ (\sigma_i \text{ a Herbrand interpretation for } L) \\ (\sigma_2 \text{ a substitution for } L) \\ (\sigma_1 \text{ grounds } H') \\ (\sigma_2 \text{ grounds } H^\omega) \\ (\hat{H}' = H'\sigma_1) \\ (\hat{H}^2 = H^\omega\sigma_2) \\ (\hat{H}' \times \hat{H}^2 \models \phi_{HL}^R \hat{H}^3) \} \\
\end{array}
\]

----->

\[\exists \sigma_3 \in H^3 : \{ (H' \times H^2 \models \phi_{HL}^R \hat{H}^3) \} \]

\[\hat{H}' \models H^3\sigma_3 \}

\[\text{ proof: Besides the usual convention that no two clauses share variables, we also assume that for } i = 1, 2, \text{ } \sigma_2 \text{ is minimal in the sense that for all } \sigma_2 \text{, if } \sigma_2 \text{ is a proper subset of } \sigma_2, \text{ then } \hat{H}'_{\sigma_2} \text{ is not a ground clause.} \]

Assume the hypothesis cl-c9. All of the following are simply derivable from cl-c9:

- \( \text{ground } \hat{H}' \)
- \( \text{ (} \phi_{HL} \text{-feasible } \hat{H}' \text{) for } i = 1, 2, \text{ and } \hat{H}_f \text{ and } \hat{H}_t \text{ and ground} \)
literals \( \hat{\xi} \) and \( \hat{\iota} \) such that

\[
\begin{align*}
&\quad \left( \hat{\xi}, \hat{\eta} \quad \phi_{\hat{\eta}} \text{-false factor of } \hat{H}^{'} \right) \\
&\quad \left( \hat{\xi}, \hat{\iota} \quad \phi_{\hat{\iota}} \text{-true factor of } \hat{H}^{2} \right)
\end{align*}
\]

Thus the immediate deduction in c9 can be diagrammed as

\[
\begin{array}{c}
\hat{H}^{'} \\
\mid \\
\hat{H}_{\xi} \\
\mid \\
\hat{H}^{2}
\end{array}
\]

The plan of the proof is as follows:

Part I: To show that there exists an \( \hat{H}_{\xi} \), a \( \xi \), an \( \phi_{\hat{\xi}} \) and \( \lambda_{\hat{\xi}} \), such that

\[
\left( \hat{\xi}, \phi_{\xi} \text{-false factor of } \hat{H}^{'} \right)
\]

\[
\left( \hat{H}_{\xi} = H_{\xi} \lambda_{\hat{\xi}} \right)
\]

Part II: To show that there exists an \( \hat{H}_{\iota} \), a \( \iota \), an \( \phi_{\hat{\iota}} \) and \( \lambda_{\hat{\iota}} \) such that

\[
\left( \hat{\iota}, \phi_{\eta} \text{-true factor of } \hat{H}^{2} \right)
\]

\[
\left( \hat{H}_{\iota} = H_{\iota} \lambda_{\hat{\iota}} \right)
\]

Part III: Prove a variant of Lemma 3.2-3 where "HLR" is replaced by "Binary HLR" in c9 and c10.

Part IV: To combine Parts I, II and III to prove the Lemma.

We consider clauses to be ordered pairs of ordinary sets of (HL-) literals. Two literals are equal only if they are equal in all three components. We will use the symbol
"^\wedge" over other notational symbols to indicate the ground level. For example, to argue Part I we talk about \( \hat{U} \) and \( U \), the former being the value of \( U \) in 2.4.19.c4 at the ground level, and the latter the corresponding set at the general level.

Part I.

p1. \( \left( \hat{\mathcal{H}}_f \right) \) -false factor of \( \mathcal{H}' \) \( \iff \)

\[ \exists \hat{U} : \left( \hat{U} \subseteq \mathcal{S}(\mathcal{H}') \right) \quad \left( \hat{x} \in \hat{U} \right) \]

\[ \left( \forall k, \tilde{k} \in k : \left( k \in \hat{U} \right) \left( \tilde{k} \in \hat{U} \right) \iff \left( l(\tilde{k}) = l(\tilde{\hat{k}}) \right) \right) \]

\[ \left( \mathcal{H}_f = \mathcal{S}(\mathcal{H}') \cup \mathcal{S}(\mathcal{H}') \cup \mathcal{S}(\mathcal{H}') \right) \]

\[ \left( \forall \hat{U} : \hat{U} \in \mathcal{S}(\mathcal{H}') \iff f(\hat{x}) \leq f(\hat{U}) \right) \]

from 2.4.19.c3, 2.4.19.c4 and 2.4.19.c6.

p2. From c7 and p1 there exists an \( S_f \) in \( \mathcal{H}' \) such that

\[ \left( S_f \in \mathcal{S}(\mathcal{H}') \right) \quad \left( \hat{S}_f = S_f \sigma_f \right) \]

\[ \left( \forall \hat{U} : \hat{U} \in \mathcal{S}(\mathcal{H}') \iff f(S_f) \leq f(\hat{U}) \right) \]

and a \( U \) such that

\[ \left( S_f \in U \right) \quad \left( U \subseteq \mathcal{S}(\mathcal{H}') \right) \quad \left( \hat{U} = \hat{U} \sigma_f \right) \]

\[ \left( \forall k : \left( k \in \mathcal{S}(\mathcal{H}') \right) \wedge (k, \hat{x} \in U) \iff (k \sigma_f, \hat{x} \in \hat{U}) \right) \]

p3. From p1 and p2 we infer that (unifiable \( U \)) and that there exists both a \( \tau = \text{mgu}(U) \), and a \( \lambda_f \) such that

\[ \sigma_f = \tau \lambda_f \]

p4. Taking \( S_f, \tau, U \) as determined in p2 and p3, we construct

\[ \mathcal{H}_f = \mathcal{S}(\mathcal{H}') \cup \mathcal{S}(\mathcal{H}') \tau \]

\[ \left( \tau(\mathcal{H}') \cup \mathcal{S}(\mathcal{H}') \tau \right) \]

p5. We now show that \( \left( S_f \mathcal{H}_f \right) \) -false factor of \( \mathcal{H}' \),

by considering each of the 7 conditions in 2.4.19 in turn.
We start with 2.4.19.c7.

\[ \mathcal{F}(H_f) \lambda_f = ( (\mathcal{F}(H') \cup \mathcal{L}(H')) \sigma_f ) \lambda_f \]

but \(2.2.6 = \sigma_f\), thus

\[ \mathcal{F}(H_f) \lambda_f = ( \mathcal{F}(H') \cup \mathcal{L}(H')) \sigma_f = \]

\[ (\mathcal{F}(\hat{H}') \cup \mathcal{L}(\hat{H}')) = \mathcal{F}(\hat{H}_f). \]

But \(\hat{H}_f\) is feasible, therefore its FSL is a set of ground literals, each of which is false in \(h\). Thus \(\phi_\mathcal{H}(\mathcal{F}(H_f)) = \text{false}\), and \(H_f\) is feasible, and 2.4.19.c7 is satisfied. By the construction of \(H_f\) in p4, 2.4.19.c6 will be satisfied. By p2 and p3, 2.4.19.c4 and 2.4.19.c5 are satisfied. By p2, 2.4.19.c3 and 2.4.19.c2 are satisfied.

We now show 2.4.19.cl is satisfied. Since (ground \(\hat{H}'\)) and (\(\phi_\mathcal{H}\)-feasible \(\hat{H}'\)) and the fact that there exists a \(\phi_\mathcal{H}\)-false factor of \(\hat{H}'\), we can show, from 2.4.14 and 2.4.13 that

\[ \phi_\mathcal{H}(\mathcal{F}(\hat{H}')) = \text{false} \]

and thus \(\phi_\mathcal{H}(\mathcal{F}(H')) = \text{false}\), and therefore \(H'\) is feasible, and also that \(\phi_\mathcal{H}(\alpha(\hat{H}')) = \text{false}\). From this we can show that \(\phi_\mathcal{H}(\alpha(H')) = \text{false}\), and therefore t.v.d. \((H', \phi_\mathcal{H}) \vdash T\), and thus 2.4.19.cl is satisfied for \(H'\).

p6. Thus we have shown, by p1-p5, how each \(\hat{H}_f\) determines an \(\mathcal{S}_f, U, \tau, \lambda_f\) such that by virtue of definition 2.4.19, there exists an \(H'_f\):

\[ (\mathcal{S}_f)_{H_f} \phi_\mathcal{H}\text{-false factor of } H' \]

\[ (\hat{H}_f = (H_f) \lambda_f) \]

This completes the proof of Part I.
Part II: We do not do Part II explicitly. The proof is almost identical to Part I, except that the \( \mathcal{A} \) in 2.4.20.c3 for \( \hat{\mathcal{A}} \) must also be shown to be computable from \( \hat{\mathcal{A}} \) in 2.4.20.c3 for \( \hat{\mathcal{A}} \) (the different FSL structure in 2.4.20.c6 compared to 2.4.19.c6 causes no substantial change in the proof of Part II compared to Part I).

Part III: Define Lemma 3.2-3.V to be the variant of Lemma 3.2-3 in which c9 is replaced by

\[
\exists \hat{\mathcal{P}} \exists \hat{\mathcal{Q}} : \left( \hat{\mathcal{A}}' \times \hat{\mathcal{A}}^2 \bigg| \sum_{k=1}^{n} \frac{m_k}{m_k} \cdot \hat{\mathcal{A}}^g \bigg) \right) \quad \text{c9.V.a}
\]

(there is a unique pre-image of \( \hat{\mathcal{P}} \) in \( \mathcal{A}' \)) \( \text{c9.V.b} \)

(there is a unique pre-image of \( \hat{\mathcal{Q}} \) in \( \mathcal{A}^2 \)) \( \text{c9.V.c} \)

and c10 and c11 are replaced by

\[
\exists \mathcal{P} \exists \mathcal{Q} \exists \mathcal{G}_0 \exists \mathcal{H}^3 : \left( \mathcal{H}' \times \mathcal{H}^2 \bigg| \sum_{k=1}^{n} \frac{m_k}{m_k} \cdot \mathcal{H}^g \bigg) \right) \quad \text{c10.V}
\]

\[
(\mathcal{H}^3 - \mathcal{H}^g \mathcal{G}^2) \quad \text{c11.V}
\]

The meaning of c9.V.b is as follows. We consider the standard literals in \( \hat{\mathcal{A}}' \) to be ground images of literals in \( \mathcal{A}' \), and for \( \hat{\mathcal{H}} \in \mathcal{S}(\hat{\mathcal{A}}' \mathcal{F}) \), the set

\[
\hat{\mathcal{H}}^{pre} = \{ \mathcal{H}^* | \mathcal{H}^* \in \mathcal{S}(\mathcal{A}') \land \mathcal{H}^* \mathcal{G}_0 = \hat{\mathcal{H}} \}
\]

is the set of pre-images of \( \hat{\mathcal{H}} \) (in \( \mathcal{A}' \), under \( \mathcal{G}_0 \)). Condition c9.V.b says that \( \hat{\mathcal{P}}^{pre} \) is a singleton. Condition c9.V.c makes the analogous assertion for \( \hat{\mathcal{Q}}^{pre} \) (in \( \mathcal{A}^2 \), under \( \mathcal{G}_0 \)).
To prove Lemma 3.2-3.V, assume c1-c8, c9.V.a, c9.V.b and c9.V.c are all true. Note that $H'$ and $H^2$ are assumed standardized apart, so that $\sigma_i$ and $\sigma_x$ substitute on disjoint sets of variables.

p7. Then we have from c9.V.a and 2.4.18 used in c9.V.a,

$$
(\bar{6}_i \in S(H')) \\
(\bar{6}_x \in S(H^2)) \\
(\forall \hat{h} \in S(H') \implies f(\bar{6}_x) \leq f(\hat{h})) \\
(1(\bar{6}_x) = \sim(1(\bar{6}_x))).
$$

p8. Let $S_x$ be the unique pre-image in $S(H')$ such that $\bar{6}_x = S_x \sigma_i$ and $S_x$ be the unique pre-image in $S(H^2)$ such that $\bar{6}_x = S_x \sigma_x$. Then (unifiable $\{S_x, \sim S_x\}$), and we let $\sigma = \text{mgu}(\{S_x, \sim S_x\})$. Clearly $\sigma_i = \sigma \lambda_i$ and $\sigma_x = \sigma \lambda_x$ for some $\lambda_i, \lambda_x$.

It can be shown that

$$
H' \sigma_i = H'(\sigma(\lambda_i, \lambda_x))
$$

and

$$
H^2 \sigma_x = H^2(\sigma(\lambda_i, \lambda_x)).
$$

p9. Let $H^3$ be constructed such that

$$
S(H^3) = (S(H^3) \sigma - (S_x) \sigma) \cup (S(H') \sigma - (S_x) \sigma) \\
\varphi(H^3) = (\varphi(H^3) \cup \varphi(H')) \sigma.
$$

p10. We now show that letting $H^3, S_x, S_x$ be as defined in p8 and p9, will make

$$
H' \times H^2 \longrightarrow H^3
$$

true, by virtue of 2.4.18, and thus satisfy c10.V. This is
done by considering all of the conditions, in turn, in 2.4.18, starting with 2.4.18.c9. We intend that $\sigma$ in 2.4.18 is to be the $\sigma$ defined in p8.

$$\mathcal{F}(\mathcal{H}^3) = (\mathcal{F}(\mathcal{H}^2) \sigma \bigcup \mathcal{F}(\mathcal{H}^l)_{\sigma^-}) .$$

$$\mathcal{F}(\mathcal{H}^3) (\lambda, \bigcup \lambda_\alpha) =$$

$$= (\mathcal{F}(\mathcal{H}^2) (\sigma (\lambda_1 \bigcup \lambda_2 )) \bigcup \mathcal{F}(\mathcal{H}^l) (\sigma^- (\lambda, \bigcup \lambda_\alpha)))$$

but this last expression is a set of ground literals, each false in h. Therefore $\phi_h (\mathcal{F}(\mathcal{H}^3)) = \text{false}$, and $\mathcal{H}^3$ is feasible, satisfying 2.4.18.c9. From p9, 2.4.18.c7 and 2.4.18.c8 are obviously satisfied.

From p7 and p8, it can be shown that 2.4.18.c3 through 2.4.18.c6 are all satisfied.

To show 2.4.18.c2, note that from c10.5 and (ground $\mathcal{H}^2$) we can show that t.v.d. ($\mathcal{H}^2, \phi_h$) = T and $\phi_h (\beta (\mathcal{H}^2)) = \text{false}$. Therefore $\phi_h (\beta (\mathcal{H}^2)) = \text{false}$, and t.v.d. ($\mathcal{H}^2, \phi_h$) $\downarrow$ F, and thus 2.4.18.c2 is satisfied. Showing 2.4.18.c1 is satisfied can be done easily in a similar way.

p11. It remains to show that c11.5 is satisfied. From p8, and the construction of $\mathcal{H}^5$ in p9 it should be clear that taking $\sigma^5 = (\lambda, \bigcup \lambda_\alpha)$ will satisfy c11.5.

This completes Part III.
Part IV.

Parts I, II and III of this proof are combined in the obvious way (by the ground diagram implied by c9), to give a general level false factoring, a general level true factoring, and a general level binary step. It is only necessary to observe that, because at the ground level the selected literals for factoring are also the selected literals resolved upon, that the same thing holds at the general level, and that the unique pre-image conditions of Part III are met (in Part I this is seen by considering the construction of \( U \) in p2, specifically the last line of p2; a similar condition is to be invoked in Part II for the true factoring). Thus the lifted sequence of two factorings followed by the binary deduction step meets all of the conditions of 2.4.21, and is thus an HLR deduction step, and C10 is therefore satisfied. That C11 is satisfied also follows, from Part III (p11) of this proof.

Q.E.D.
Lemma 3.2 -- 4

(Herbrand's Theorem for NL-Clauses)

\[
\exists \mathcal{H} : \forall \overrightarrow{\mathcal{H}} ( \overrightarrow{\mathcal{H}} \in \mathcal{H}) \\
\exists \mathcal{H} \exists \sigma ( \mathcal{H} \in \mathcal{H}) \\
(\sigma \text{ a substitution of } \mathcal{L}) \\
(\overrightarrow{\mathcal{H}} = \mathcal{H}\sigma) \\
(\text{ground } \overrightarrow{\mathcal{H}}) \\
\text{(un satisfiable } \mathcal{H}) \\
\text{ (} \mathcal{H} \text{ is a finite set) .}
\]

proof: Immediate from Herbrand's theorem applied to \(\mathcal{L}(\mathcal{H})\) and the various definitions of unconstrained, unsatisfiable, and \(\phi_{\mathcal{H}}\).

We are now ready to prove the refutation completeness of NL-resolution at the general level when the semantic function is the usual Herbrand semantic function, \(\phi_{\mathcal{H}}\). We repeat the statement of the theorem here.
Theorem 3.1 -- 2
(Herbrand Model Completeness of HLR)

\[
\begin{align*}
\text{(ML-proper } \mathcal{H}) & \quad c_1 \\
\text{(unconstrained } \mathcal{H}) & \quad c_2 \\
\text{(unsatisfiable } \mathcal{H}) & \quad c_3 \\
\text{(L a language for } \mathcal{H}) & \quad c_4 \\
\text{(h a Herbrand interpretation for L) } & \quad c_5
\end{align*}
\]

\[\implies\]

\[\exists \mathcal{H} \text{ null } \mathcal{H} \quad c_6
\]

\[
\begin{align*}
\text{( } \mathcal{H} \text{ is HLR refutation of } \mathcal{H}) & \quad c_7
\end{align*}
\]

**proof:** Assume c1 - c5 are true.

By c2, c3 and Lemma 3.2-4 there exists a set \(\mathcal{H}'\), which is unsatisfiable, and consists of ground instances of clauses in \(\mathcal{H}\). By Theorem 3.2-2 and c1 - c5, we have

\[\exists \mathcal{H} \text{ null } \mathcal{H} \quad c_6
\]

\[
\begin{align*}
\text{( } \mathcal{H} \text{ is HLR refutation of } \mathcal{H}) & \quad c_7
\end{align*}
\]

Therefore there exists an \(\hat{\mathcal{H}}\) such that \(\hat{\mathcal{H}}\) is a ground HLR refutation of \(\mathcal{H}'\) using \(\phi_{\mathcal{H}}\). We transform \(\hat{\mathcal{H}}\) as follows.

Transform the leaves of \(\hat{\mathcal{H}}\) by replacing each with a clause in \(\mathcal{H}\) of which it is an instance. Then apply Lemma 3.2-3 to successive levels of the tree to transform it into a general deduction tree, \(T\). The root will still be a null clause, therefore the tree \(T\) is an HLR refutation of \(\mathcal{H}\).

Q.E.D.
3.3 Completeness of the Inference Rule HLR - t.v.d.

In this section a refinement called HLR - t.v.d. is defined and proved complete.

We define HLR - t.v.d. to be the inference rule identical to HLR except that in definitions 2.4.18, 2.4.19 and 2.4.20, the conditions requiring particular t.v.d. values are ignored, and "HLR" is replaced by "HLR - t.v.d." in definitions 2.4.18 through 2.4.22. In each inference step of the rule HLR - t.v.d. one parent still plays the role of the false parent, and the other the role of the true parent, but each parent may have any t.v.d. value. Trivial variations of Theorems 2.5-1 through 2.5-3 give the soundness of HLR - t.v.d. using any semantic function.

If \( \psi = \phi_h \) for some Herbrand interpretation, \( h \), for the language of \( \mathcal{L} \), then HLR - t.v.d. is clearly a refutation complete refinement of resolution (from Theorem 3.1-2 and the fact that every HLR refutation using \( \phi_h \) is an HLR - t.v.d. refutation using \( \phi_h \)). The next theorem shows that, when using \( \phi_h \), HLR and HLR - t.v.d. are identical refinements.
Theorem 3.3 -- 1

(t.v.d.'s are superfluous in HLR using $\phi_A$)

\begin{align*}
&1 \text{ (HL-proper $\mathcal{M}$)} \quad \text{c1} \\
&2 \text{ (L a language for $\mathcal{M}$)} \quad \text{c2} \\
&3 \text{ (h a Herbrand interpretation for L)} \quad \text{c3} \\
&4 \text{ (R is an HLR - t.v.d. deduction of } \mathcal{M} \text{ from $\mathcal{M}$ using $\phi_A$) } \quad \text{c4}
\end{align*}

\[ \rightarrow \]

\[ \text{[R is an HLR deduction of } \mathcal{M} \text{ from $\mathcal{M}$ using $\phi_A$] } \quad \text{c5} \]

proof: By contradiction. Assume c1 - c4 true and c5 false.

Then there exists nodes $H^1, H^2, H^3$ of R such that

\[ H^1 \times H^2 \xrightarrow{\phi_A} H^3 \]

is an immediate HLR - t.v.d. deduction which is a subtree of R, but it is not the case that $H^1 \times H^2 \xrightarrow{\phi_h} H^3$, (we ignore the trivial case of $H \in \mathcal{M}$).

Since HLR and HLR - t.v.d. differ only in the t.v.d. conditions for clauses, it must be the case that at least one clause has a t.v.d. value that blocks the HLR inference step. We expand the HLR - t.v.d. inference step to

\[
\mathcal{N} \in \begin{cases} 
H^1 \\
H^2 \\
H^3
\end{cases}
\]
where \( H_f \) is a false factor of \( H' \) and \( H_h \) is a true factor of \( H^2 \), and \( H^3 \) is an HLR - t.v.d. binary resolvent of \( H_f \) against \( H_h \). Let \( S_f \) and \( S_h \) be the selected literals for factoring in \( H' \) and \( H^2 \), respectively, and the corresponding substitutions used in the factoring be \( \tau \) and \( \varphi \). Then the selected literals for the binary step are \( (S_f)\tau \) and \( (S_h)\varphi \). Let the unifier in the binary HLR - t.v.d. step be \( \sigma \). Clearly there are 4 ways for t.v.d. values to be wrong for HLR, namely,

Case 1. t.v.d.\((H', \phi_h) = T\)
Case 2. t.v.d.\((H^2, \phi_h) = F\)
Case 3. t.v.d.\((H_f, \phi_h) = T\)
Case 4. t.v.d.\((H_e, \phi_h) = F\).

We show that a contradiction is obtainable if any (one or more) of the above 4 conditions hold.

Case 1. t.v.d.\((H', \phi_h) = T\).
Because of \( \mathcal{H} \), we have \( H_f \) is \( \phi_h \)-feasible. But
\[
\mathcal{F}(H_f) = (\mathcal{S}(H') \cup \mathcal{F}(H')) \tau.
\]
Thus \( \phi_h(\mathcal{S}(H') \cup \mathcal{F}(H')) = false \).
Therefore \( \phi_h(\mathcal{S}(H')) = false \), and t.v.d.\((H', \phi_h) \not\subset T\).
Thus we have the contradiction in this case.
Case 2. $t.v.d.\ (H^2, \phi_k) = F$.

From $\mathcal{H}$ we have $H_\xi$ is $\phi_k$-feasible, thus
\[ \phi_k(\mathcal{N}(H_\xi)) = \text{false}, \] and therefore there exists a $\lambda$:

\[ \lambda \text{ grounds } H_\xi \]
\[ \mathcal{N}(H_\xi) \lambda \text{ consists only of false ground literals} \]
\[ \mathcal{N}(H_\xi) = (\mathcal{N}(H^2) \cup S_\xi \cup \Omega) \phi \]

where $\Omega$ is a subset of $\mathcal{S}(H^2)$. Therefore there exists a $\lambda'$:

\[ \lambda' \in \mathcal{S}(H^2) \]
\[ \lambda' \text{ grounds } H^2 \]
\[ \mathcal{N}(H^2) \lambda' \text{ consists only of false ground literals} \]
\[ \text{ground } (S_\xi) \lambda' \]
\[ (S_\xi) \lambda' \in \mathcal{S}(H^2) \lambda' \]
\[ \phi_k((S_\xi) \lambda') = \text{true}. \]

This then implies that $\phi_k(\beta(H^2)) = \text{false}$. Therefore $t.v.d.\ (H^2, \phi_k) \not\vdash F$, and we have the contradiction.

Case 3. $t.v.d.\ (H_\xi, \phi_k) = T$.

From definition 2.4.19 and the fact that $H_\xi$ is a false factor (of $H'$), we know that $L(\mathcal{S}(H_\xi)) \subseteq L(\mathcal{N}(H_\xi))$. But from $\mathcal{N}$, $\phi_k(1(\mathcal{N}(H_\xi))) = \text{false}$, therefore $\exists \lambda : [ (\lambda \text{ a substitution of } L) \]

\[ \lambda \text{ grounds } H_\xi \]
\[ \phi_k(1(\mathcal{N}(H_\xi)) \lambda) = \text{false}. \]

Therefore $\phi_k(1(\mathcal{S}(H_\xi)) \lambda) = \text{false}$.

Thus $\phi_k(\alpha(H_\xi)) = \text{false}$, and $t.v.d.\ (H_\xi, \phi_k) \not\vdash T$
and we have the contradiction.
Case 4. \( t.v.d. (H_t, \phi_k) = F \).

From \( \mathcal{N} \), \( H_t \) is a \( \phi_k \)-true factor (of \( H^2 \)), and is \( \phi_k \)-feasible.

Therefore \( \exists \lambda : [ (\lambda \text{ a substitution of } L) \)

\( (2 \text{ grounds } H_t) \)

\( (\phi_k(\neg(\neg H_t)\lambda) = \text{false}) ] \).

Therefore ground \( \neg S_c \in \lambda \), from 2.4.20.c6.

\( \phi_k(\neg S_c \in \lambda) = \text{false} \)

\( \phi_k(S_c \in \lambda) = \text{true} \).

But \( S_c \in S(H_t) \). Therefore \( \phi_k(S(H_t)) = \text{false} \).

Therefore \( t.v.d.(H_t, \phi_k) \not\in F \), and we have the contradiction in this case.

Q.E.D.

Theorem 3.3-1 shows that the t.v.d. conditions in the HLR inference rule using \( \phi_k \) are completely redundant with respect to the feasibility conditions on FSL's. This then implies that HLR and HLR - t.v.d. are the same identical refinement when the semantic function is \( \phi_k \). In an actual implementation the use of the t.v.d. conditions in HLR does serve a purpose however, in that many potential parent pairs can be excluded on the basis of t.v.d. values instead of having to produce the FSL of the resolvent and test it for feasibility. A substantial amount of work can be saved by the use of the t.v.d. 's, and this is why HLR includes them, even when using a Herbrand semantic function. For arbitrary semantic functions, even if they are sound, it is not the
case that HLR and HLR - t.v.d. are identical refinements, nor is it necessarily the case that one will be a refinement of the other.

The next theorem shows the false permissive completeness of HLR - t.v.d. using any sound semantic function.

**Theorem 3.3 - 2**

(False Permissive Completeness of HLR - t.v.d.)

\[
\{ \text{HL-proper } \mathcal{H} \} \\
\text{(unconstrained } \mathcal{H} \} \\
\text{(L a language of } \mathcal{H} \} \\
\text{ (} \Psi \text{ a sound semantic function for } L \} \\
\text{(unsatisfiable } \mathcal{H} \} \\
\}\rightarrow \\
\{ \exists \phi : \mathcal{H} \text{ (null } \mathcal{H} \} \\
\text{( } \mathcal{H} \vdash \phi \text{ false permissive wrt. } h \} \\
\}\]

**proof:** By c4 there exists an h such that

(h a Herbrand interpretation for L)

(\(\Psi\) false permissive wrt. h).

By the Herbrand model refutation completeness of HLR - t.v.d., there exists an HLR - t.v.d. refutation, R, of \(\mathcal{H}\) using \(\phi_h\). But the only way that the HLR - t.v.d. inference rule depends upon a semantic function is in the
feasibility conditions. Every clause in $R$ is $\phi_A$-feasible, and therefore is $\psi$-feasible (this also includes the factors of clauses involved in each inference step). Thus the existence of $R$ and its root satisfy conditions $c6$ and $c7$.

Q.E.D.

If Theorem 3.3-1 held with $\phi_A$ replaced by any sound semantic function, $\psi$, then this along with Theorem 3.3-2 would establish the false permissive completeness of HLR. However, Theorem 3.3-1 does not hold with $\phi_A$ replaced by an arbitrary sound $\psi$. The reason is that the proof of Theorem 3.3-1 uses properties of $\phi_A$ which are not necessarily true for a particular sound $\psi$. For example, if $K$ and $K'$ are sets of literals (in the appropriate language) and $K \subseteq K'$, then

$$\phi_A(K) = \text{true} \implies \phi_A(K') = \text{true},$$

but for some arbitrary sound $\psi'$, it is not necessarily true that

$$\psi(K) = \text{true} \implies \psi(K') = \text{true},$$

(even when ($\psi$ false permissive wrt. $\phi_A$)). In Chapter 4 a methodology for defining and implementing sound semantic functions is given which can yield in practice sound semantic functions which will make HLR deduction incomplete with respect to HLR - t.v.d. Thus the false permissive completeness of HLR must be established by other means.
3.4 MLA False Permissive Completeness Proof.

This section is the proof of Theorem 3.1-1, which is restated below. Note that this proof does not use any characteristics of $\mathcal{V}$ beyond it being a sound semantic function.

We will need the following lemma.

Lemma 3.4 -- 1

(A t.v.d. Conservation Property)

\[
\begin{align*}
( & \text{L a language for } \{ \mathcal{H}, \mathcal{L} \} ) \quad \text{c1} \\
( & \text{h a Herbrand interpretation for } \text{L} ) \quad \text{c2} \\
( & \text{false permissive wrt. } \phi_k ) \quad \text{c3} \\
\implies \quad \text{c4} \\
\implies \quad \text{c5}
\end{align*}
\]

\[
\begin{align*}
( & (t.v.d.((\mathcal{H}, \phi_k) \downarrow T) \\
\implies \quad \text{c4} \\
\implies \quad \text{c5}) \\
\implies \quad \text{c5}) \quad \text{c4} \\
\implies \quad \text{c5}) \quad \text{c5}
\end{align*}
\]

proof: Assume c1 - c3 are true. We will explicitly argue only c4. The proof of c5 is similar. Assume

\[t.v.d.((\mathcal{H}, \phi_k) \downarrow T),\]

i.e.

\[t.v.d.((\mathcal{H}, \phi_k) = F \lor t.v.d.((\mathcal{H}, \phi_k) = T/F.
\]

We now show that t.v.d.((\mathcal{H}, \mathcal{V}) is either F or T/F.

Case 1. t.v.d.((\mathcal{H}, \phi_k) = F.

Then from definition 2.4.14
\[ \phi_A( \models (H \vdash *)) = \text{false} \]
\[ \phi_A( \beta (H \vdash *)) = \text{true} \].

Since \( \Psi \) is false permissive with respect to \( \phi_A \), we will have

\[ \Psi( \models (H \vdash *)) = \text{false} \].

Case 1.1. \( \Psi( \beta (H \vdash *)) = \text{true} \). Then \( \text{t.v.d.}(H \vdash, \Psi) = F \).

Case 1.2. \( \Psi( \beta (H \vdash *)) = \text{false} \).

Then \( \text{t.v.d.}(H \vdash, \Psi) = T/F \).

Case 2. \( \text{t.v.d.}(H \vdash, \phi_A) = T/F \).

Thus \( \phi_A( \models (H \vdash *)) = \text{false} \) and \( \phi_A( \beta (H \vdash *)) = \text{false} \).

Therefore
\[ \Psi( \models (H \vdash *)) = \text{false} \]
\[ \Psi( \beta (H \vdash *)) = \text{false} \]
and \( \text{t.v.d.}(H \vdash, \Psi) = T/F \).
Thus the t.v.d. of \( H \vdash \) under \( \Psi \) is F or T/F.

Q.E.D.

Theorem 3.1 -- 1
(False Permissive Completeness of HLR)

\[
\begin{array}{c}
(\text{HL-proper } \mathcal{K}) \\
(\text{unconstrained } \mathcal{K}) \\
(\text{unsatisfiable } \mathcal{K}) \\
(L \text{ a language for } \mathcal{K}) \\
(\Psi \text{ a sound semantic function for } L)
\end{array}
\]

\[
\begin{array}{c}
\exists \mathcal{M} \left[ \text{null } \mathcal{M} \right] \\
(\mathcal{K} \vdash_{\text{H-L}} \Psi \rightarrow \mathcal{M}) \]
\end{array}
\]
proof: By c5 there exists an h such that
(h a Herbrand interpretation for L)
(Ψ false permisive wrt. φh).

By the refutation completeness of HLR using φh, (Theorem 3.1-2), there exists an HLR refutation, R, of Ψ, using φh. The two ways that the HLR inference rule depends upon the semantic function is through the feasibility conditions and the t.v.d. conditions on parent clauses. Every clause in R is φh-feasible, and is therefore Ψ-feasible. Likewise all of the factors involved in each immediate deduction subtree in R are Ψ-feasible.

Thus, if it can be shown that for all immediate deduction subtrees contained in R the t.v.d. values are correct using Ψ instead of φh, then R will be a refutation satisfying conditions c6 and c7.

Let

\[ \mathcal{N} \subseteq \mathcal{H}_f \times \mathcal{H}_i \mid_{\phi_h} \mathcal{H}_k \]

be an arbitrary immediate deduction subtree of R.

Let \( \mathcal{H}_f \) and \( \mathcal{H}_b \) be the \( \phi_h \)-false and \( \phi_h \)-true factors, respectively, of \( \mathcal{H}_i \) and \( \mathcal{H}_k \) involved in \( \mathcal{N} \). Then the immediate deduction \( \mathcal{N} \) can be expanded as
To show that this is an immediate HLR deduction using $\psi$ requires that it be shown that the following all hold:

\begin{align*}
  &i. \ t \ v \ d. (\neg \neg \phi, \psi) \vdash T \\
  &ii. \ t \ v \ d. (\neg \phi, \psi) \vdash T \\
  &iii. \ t \ v \ d. (\neg \phi, \psi) \vdash F \\
  &iv. \ t \ v \ d. (\neg \phi, \psi) \vdash F.
\end{align*}

But since $\mathcal{H}$ is an HLR immediate deduction using $\phi$, we know that the above 4 conditions all hold with $\phi$ replacing $\psi$. Then by Lemma 3.4-1 the conditions also hold as written above. Thus $\mathcal{H}$ is also an immediate deduction according to HLR using $\psi$. Thus $R$ is an HLR refutation of $\mathcal{H}$ using $\psi$, and conditions c6 and c7 are satisfied.

Q.E.D.

This concludes the treatment of the completeness of HLR. We defer the general discussion of HLR until Chapter 5, so that the material on models in Chapter 4 will be available.
Chapter 4

MODELS

While there already exists a large and growing theory of models for first order logic (see, e.g., [Chang and Keisler, 1973]), there is a notable lack of application of model theory to resolution refinement strategies (with the exceptions of [Manschen, 1975], and [Brown, 1973], both of which are restricted to Horn sets) beyond the initial development of the model strategy of Luckham and the semantic resolution strategy of Slagle.

It has been pointed out (Kowalski and Hayes, 1969) that there are in particular two problems arising in connection with the use of interpretations in resolution theorem proving:

1. the interpretation must be able to assign truth values to clauses which contain Skolem functions which are not part of the original language of the domain, but rather are introduced by the process of converting the negation of the theorem into clauses;
2. there must be an effective way to compute the truth values of clauses relative to the interpretation.

These two problems are closely related, and they both have their solution in the way in which the interpretation is specified. It is the purpose of this chapter to present a paradigm and methodology of how to specify and use
sophisticated models in resolution strategies such as TMS and HLR, and not suffer from the two difficulties above.

It was proven in Chapter 3 that HLR is a false permissive complete refinement. Most other semantic refinements of resolution are also false permissive complete. Thus we will be concerned with developing a methodology for specifying sound semantic functions. We list here some overall aspects of the approach taken in this chapter:

1. The sound semantic functions are semantic functions for a language, which we will call the language of the clauses, $L_c$, and these functions need not have any particular properties relative to the input clauses beyond that implied by the fact that each clause is a sentence in $L_c$.

2. The approach retains the emphasis on Herbrand interpretations and the induced relational structures of Herbrand interpretations, as opposed to arbitrary relational structures, since the Herbrand interpretations are more appropriate and familiar.

3. In general, the term model will refer to structures, entities, or procedures, which are not the usual relational structures of first order model theory (one example of this has already been used in this thesis, namely defining a Herbrand interpretation to be a set of literals, and then saying that this induces a relational structure, as in section 2.4.8.6), and we will be
concerned with defining (sound) semantic functions based on these structures, entities or procedures.

4. An additional language, besides \( L_c \), will be involved in the methodology; this language is called the language of the model, \( L_m \), and is employed mainly for heuristic value in thinking about models, and for purposes of exposition, and it is intended that a special case of the approach be when \( L_m = L_c \).

5. Several transitions (of the characteristics of models, how they are specified, and their associated semantic functions) will occur in the course of the chapter and include in particular the following:

   a. from static, declarative and theoretical notions of interpretation (as in Herbrand interpretations) to computational, pragmatic and procedural notions of interpretation;

   b. from \( \phi \) as the semantic function to sound semantic functions which are not equal to any Herbrand semantic function;

   c. from specifying complete interpretations to partial specification of interpretations;

   d. from giving the truth values of ground literals as the means of specifying a model, to giving the relationships among non-ground literals.
6. Throughout the chapter the role of the relation "false permissive wrt." is crucial to the methodology, both from the abstract viewpoint of establishing the lemmas and theorems of this chapter, and from the viewpoint of implementing in pragmatic situations the sound semantic functions which the methodology leads to.

The treatment of models of this chapter is an expanded version of material available in (Sandford, 1977b). The reader is encouraged to read (Reiter, 1972) for a different treatment of models in a natural inference system context.

A discussion of the rationale for using models in the first place will be deferred until the conclusion of this chapter, where it can be treated in a more adequate context using the concepts presented in the body of this chapter. It would not be inappropriate for the reader to quickly read section 4.6 at this time to obtain a rough notion of the end product of the lengthy exposition of this chapter.
4.1 Notation and Preliminary Definitions.

Section 4.1 contains definitions of the basic notions which will be used in later sections.

4.1.1 Language Conventions. We will use generally the same meta-language conventions as used in defining NLR. L will be used as a generic symbol for first order languages. We will be concerned with three first order languages in the context of a first order resolution modeling situation.

The first is the language of the clauses, $L_C$, and is the language in which the set of clauses is written and in which the proof is sought.

The second is the language of the model, $L_M$, and is the language in which the model is specified in an abstract sense. $L_M$ is not necessarily explicitly present in an implementation of the model, although in some cases it could be. On the other hand, $L_C$ is always explicitly present, since the search for a proof involves generating clauses in $L_C$.

The third language, denoted by $L_C \setminus L_M$, is a reduction of $L_M$, and will be defined later. $L_C \setminus L_M$ is of interest only in proving that certain relationships hold between models for $L_C$ and models for $L_M$, and never appears explicitly in an implementation (except in the trivial sense in that it is a subset of $L_M$).
$L_c$, $L_m$, and $L_c \setminus L_m$ are all first order languages of the usual type (e.g., see (Chang and Lee, 1973), (Kleene, 1967), or almost any other text on first order logic) with the following minor changes:

1. there is a single set (infinite) of variable symbols shared by all languages;
2. in matters concerning models, any lock numbers on literals are invisible, so that a set of HL-literals, $K$, is equivalent to $1(K)$;
3. the conventions of section 2.4.5 are retained, so that we can speak of a normal clause (i.e. a set of literals) as being a sentence in a language;

For purposes of exposition we will treat explicitly only cases where the relation and function symbols of $L_c$ are all distinct from those of $L_m$.

We do not assume that there is automatically a distinguished relation symbol in our languages standing for the notion of equality.

A language can be specified by listing its relation and function symbols (constant symbols are zero place function symbols). We define a function, desc, mapping languages to their description:

$$\text{desc}(L) = < \Delta(L), \Gamma(L) >$$

where $\Delta$ maps $L$ to its set of relation symbols, and $\Gamma$ maps $L$ to its set of function symbols. It is assumed that $\Delta(L)$ is not empty and that $\Gamma(L)$ contains at least one constant
symbol.

We let \textit{arity} be a function mapping relation and function symbols of a language to their degree. For example, if \( c \) is a constant symbol of a language \( L \), we have

\[ \text{arity}(c,L) = 0. \]

The \textit{alphabet} of a language, \( L \), is the union of

1. \( \Delta(L) \)
2. \( \Gamma(L) \)
3. logical operators (e.g. "\&", "\wedge", etc.)
4. variable symbols (\( x, y, z, \ldots x', \ldots \))
5. quantifier symbols ("\( \forall \)" and "\( \exists \)"")
6. various punctuation ("", ", :", ) and grouping ("(" , ")", "[", "]") symbols (Punctuation symbols are for readability only, and have no other meaning).

4.1.2 Amicability.

4.1.2.1 (\( \forall \) an amicability function from \( L \) to \( L' \))

iff

\[
\begin{align*}
( \forall \text{ is total on } \Delta(L) \cup \Gamma(L) ) & \\
( \forall \in: \left\{ \begin{array}{l}
( s \in \Delta(L) \implies \forall(s) \in \Delta(L') ) \\
( s \in \Gamma(L) \implies \forall(s) \in \Gamma(L') ) \\
( s \in (\Delta(L) \cup \Gamma(L)) \\
\implies \text{arity}(s,L) = \text{arity}(\forall(s),L') \end{array} \right\} ) \end{align*}
\]

Notice that \( \forall \) may be many-one, and need not be onto \( \Delta(L') \cup \Gamma(L') \).
4.1.2.2 \((L' \text{ amicable to } L)\)

iff

\[ \exists \psi : (\psi \text{ an amicability function from } L \text{ to } L') \].

4.1.3 \text{Translation Function.}

Suppose \((L' \text{ amicable to } L)\) and \((\psi \text{ an amicability function from } L \text{ to } L')\). Let \(\mathcal{T}\) be a total function on \(L\) into \(L'\), called a \text{translation function}, defined as follows:

i. for \((s \in \Delta(L) \cup \Gamma(L))\),
\[ \mathcal{T}(s) = \psi(s). \]

ii. for \((s \notin \Delta(L) \cup \Gamma(L))\)
\(s \in \text{alphabet of } L\),
\[ \mathcal{T}(s) = s. \]

iii. for \(K = s_1 \ldots s_m\) a string of alphabet characters of \(L\),
\[ \mathcal{T}(K) = \mathcal{T}(s_1) \mathcal{T}(s_2) \ldots \mathcal{T}(s_m). \]

In such a case we say that \((\mathcal{T} \text{ translates } L \text{ to } L')\) and that \(\psi\) induces \(\mathcal{T}\).

\(\mathcal{T}\) has its domain extended in the obvious way:

If \(\mathcal{T}(k_i)\) is defined for each \(k_i \in K\), then
\[ \mathcal{T}(K) = \{ \mathcal{T}(k_i) | k_i \in K \}. \]

Note that, if \(\mathcal{N}\) is considered as a sentence, then
\[ \neg \mathcal{T}(\neg \mathcal{N}) \text{ is the same as } \mathcal{T}(\neg \mathcal{N}). \]
Thus $T$ can be applied to sets of statements and to Herbrand interpretations. In order to make $T$ applicable to substitutions we define $T(/) = /$.

It will be clear later that restricting $T$ to have such a simple form is no restriction on the power of the methodology.

4.1.4 Model Scheme.

We use $\mathcal{M}$ as a generic symbol for a model scheme. Previously $\mathfrak{M}$ was used to designate a model as an ill-defined notion. We now make the commitment that our models are really based on model schemes, which are defined next as definite set theoretic objects. Notice however that we still have not specified how the model scheme is to determine a semantic function.

A model scheme for $L$ is defined to be any non-empty subset of the set of all Herbrand interpretations for $L$. Thus we have

$$(\mathcal{M} \text{ a model scheme for } L) \iff$$

$$[(\mathcal{M} \subseteq \{ h \mid (h \text{ a Herbrand interpretation for } L) \}) \land (\mathcal{M} \nvdash \{ \})].$$

The model scheme is the fundamental notion on which the material of this chapter is based. The remaining definitions of section 4.1 show how, by specifying a theory in $L_{\mathcal{M}}$, one may induce a model scheme in $L_{\mathcal{M}}$ and a model
scheme in $L_c$. In sections 4.2 and 4.4 it will be shown how sound semantic functions for $L_c$ can be defined, and evaluated, in the environment of $L_m$. In order to do this we need next the definition of a model structure.

4.1.5 Model Structure. A model structure, $M$, is defined to be a 4-tuple:

$$M = < L_c, L_m, T, \Sigma >$$

meeting the following 4 conditions:

1. $L_c$ and $L_m$ are first order languages.
2. ($L_m$ amicable to $L_c$).
3. ($T$ translates $L_c$ to $L_m$).
4. $\Sigma$ is a theory of $L_m$ such that there exists a Herbrand interpretation, $h$, of $L_m$ such that each sentence in $\Sigma$ is true in the relational structure induced by $h$.

$L_c$ is called the language of the clauses and $L_m$ the language of the model, of $M$. $T$ is called the translation function for $M$, and $\Sigma$ is called the scheme defining set of $M$. Note that one way in which condition 4 above can be met is to have $\Sigma$ as a satisfiable set of normal clauses in $L_m$. Also note that $\Sigma$ may be infinite, and is not necessarily closed (under logical consequence).

It is the relationships between the 4 components of a model structure which will allow the sound semantic functions for $L_c$ to be defined and evaluated in $L_m$. 
For definitions 4.1.6 through 4.1.10 we assume \( L_c, L_m, \Sigma \) and \( \Sigma \) are the components of some model structure \( M \). See Figure 4.1, for a diagram of the following definitions and relationships.

4.1.6 The model scheme \( M_{\Sigma}^{L_m} \):

\[
M_{\Sigma}^{L_m} = \{ h_m \mid (h_m \text{ a Herbrand interpretation for } L_m) \}
\wedge (\phi_{m}(\Sigma) = \text{true})
\].

4.1.7 The language \( L_c \setminus L_m \):

\[
\text{desc}(L_c \setminus L_m) = \langle \mathcal{T}(\Delta(L_c)), \mathcal{T}(\Sigma(L_c)) \rangle.
\]

This is equivalent to saying that \( L_c \setminus L_m = \mathcal{T}(L_c) \).

Clearly \( L_c \setminus L_m \) is a reduction of \( L_m \).

4.1.8 The model scheme \( M_{\Sigma}^{L_c \setminus L_m} \):

\[
M_{\Sigma}^{L_c \setminus L_m} = \{ h_{c \setminus L_m} \mid
\text{(h}_{c \setminus L_m} \text{ a Herbrand interpretation for } L_c \setminus L_m) \}
\wedge \exists h_m : [ (h_{c \setminus L_m} \subseteq h_m) \wedge (h_m \in M_{\Sigma}^{L_m}) ]
\}. \]

4.1.9 The model scheme \( M_{\Sigma}^{L_c} \):

\[
M_{\Sigma}^{L_c} = \{ h_c \mid (h_c \text{ a Herbrand interpretation for } L_c) \}
\wedge (\mathcal{T}(h_c) \in M_{\Sigma}^{L_c \setminus L_m})
\).

By definition \( \Sigma \) is satisfiable, so that \( M_{\Sigma}^{L_m} \), \( M_{\Sigma}^{L_c \setminus L_m} \) and \( M_{\Sigma}^{L_c} \) are all non-empty.
4.1.10 The (model scheme) evaluation functions (called
\( \Theta \)-functions)

\( \Theta_{L^c}, \Theta_{L^c \setminus L_m}, \Theta_{L_m} \)

for the model schemes

\( M_{L^c}, M_{L^c \setminus L_m}, M_{L_m} \)

respectively, such that for a sentence, \( s \), in the language
\( L \), we have

\[
\Theta_{L^c}(s) = \begin{cases} 
  f & \text{if } \exists h_L : ( h_L \in M_{L^c} ) \\
  t & \text{otherwise}
\end{cases}
\]

\( \wedge ( \phi_{L^c}(s) = \text{false} ) \)

for \( L = L_c, L_c \setminus L_m, \) or \( L_m \).

We use \( "f" \) and \( "t" \) as the range of \( \Theta \)-functions to
suggest \( "false" \) and \( "true" \), but these values are distinct
from the \( false \) and \( true \) values of semantic functions and
from the \( T, F, T/F \) values of the t.v.d. function.

Henceforth we assume that some model structure

\( M = < L_c, L_m, T, \mathcal{L} > \)

has been selected so that the items defined in definitions
4.1.6 through 4.1.10 are all well defined.
\[ L_c \quad \mathcal{I}(L_c) = L_c \setminus L_m \quad L_c \setminus L_m \quad \text{--- reduction} \quad \text{--- expansion} \quad L_m \]

**ALL HERBRAND INTERPRETATIONS**
- FOR \( L_c \)
- FOR \( L_c \setminus L_m \)
- FOR \( L_m \)

**MODEL SCHEMES**

\[ M = < L_c, L_m, \mathcal{I}, \Sigma > \] induces \( m^L_m \),
which induces \( m^{L_c \setminus L_m} \), which induces \( m^L_c \),
which allows sound semantic functions to be defined for \( L_c \).

The scheme defining set of \( M \).

**Figure 4-1**

RELATIONSHIP OF HERBRAND INTERPRETATIONS AND MODEL SCHEMES IN DIFFERENT LANGUAGES.
4.2 Theoretical Construction of Sound Semantic Functions.

In this section the basic lemmas and theorems are presented which allow sound semantic functions to be defined for $L_e$. The treatment of this section is not addressed to the pragmatics of actually evaluating the semantic functions defined, but rather just to presenting the abstract possibilities for definition of these functions. In section 4.4 these results are extended to take into account the complexity of model evaluations in practical situations.

The reader is again reminded that $L_e$, $L_m$, $T$ and $\Sigma$ are implicitly all components of a model structure. The presentation begins by proving some properties of model schemes and $\Theta$-functions and then uses these properties to define some sound semantic functions.
Lemma 4.2

If $C$ is a set of ground literals in the language $L_c$, and $h_{lc} \in \mathcal{M}_L^{lc}$, then

$$(C \cap h_{lc} = \{\}) \implies (\mathcal{T}(C) \cap \mathcal{T}(h_{lc}) = \{\})$$

proof: Let $\mathcal{T}(h_{lc}) = h$. By definitions 4.1.9 and 4.1.8 $h_{lc} \in \mathcal{M}_L^{lc}$ implies $h$ is a Herbrand interpretation for $L_c \setminus L_m$. The proof is by contradiction. Assume $C \cap h_{lc} = \{\}$ and $\mathcal{T}(C) \cap h \neq \{\}$. Then there exists a literal, $k$, in $C$, such that $\mathcal{T}(k) \in h$, and $k \not\in h_{lc}$. Thus $\neg k \in h_{lc}$, and by definition 4.1.3 $\mathcal{T}(\neg k) \in h$. But if $\mathcal{T}(k)$ and $\mathcal{T}(\neg k)$ are both in $h$, then $h$ cannot be a Herbrand interpretation, and we have the contradiction.

Q.E.D.
Lemma 4.2-2

Let C be a set of ground literals in Lc.

If \( \exists h_{c_c} : [ (h_{c_c} \in M^L_c) \quad \text{and} \quad (\phi_{h_{c_c}}(C) = \text{false}) ] \)  

then \( \exists h_{\mu_m} : [ (h_{\mu_m} \in M^L_m) \quad \text{and} \quad (\phi_{h_{\mu_m}}(T(C)) = \text{false}) ] \)

proof:

p1. Assume c1, c2 true.

p2. Let \( h = T(h_{c_c}) \), where \( h_{c_c} \) "makes" c1 and c2 true.

p3. By definition 4.1.9 we have \( h \in M^h_{\varphi} \).

p4. \( C \cap h_{c_c} = \emptyset \), by p1 and definition 2.4.8.2.

p5. \( T(C) \cap h = \emptyset \), by Lemma 4.2-1, p2 and p4.

p6. By definition 4.1.8 and p3, there exists an \( h_{\mu_m} \) such that \( (h \subseteq h_{\mu_m}) \) and \( (h_{\mu_m} \in M^h_{\varphi}) \).

p7. From definition 4.1.6 and p6,

\( (h_{\mu_m} \) a Herbrand interpretation for \( L_m \)).

p8. \( x \in T(C) \implies (\neg x) \in h \), from p5.

p9. \( x \in T(C) \implies (\neg x) \in h_{\mu_m} \), by p6.

p10. \( x \in T(C) \implies x \notin h_{\mu_m} \), from p7, and p9.

p11. \( \phi_{h_{\mu_m}}(T(C)) = \text{false} \), from p10 and definition 2.4.8.2.

The lemma conditions c3 and c4 are satisfied by virtue of p6 and p11.

Q.E.D.
Lemma 4.2 -- 3

Let C be a set of literals in Lc. Then

\( (\bar{\theta}_c^l(C) = f) -\implies (\bar{\theta}_c^{ln}(T(C)) = f). \)

**proof:**

p1. From definitions 4.1.10 and 2.4.8.3 we have

\( \bar{\theta}_c^l(C) = f \implies \exists h \exists \sigma : \begin{array}{l}
\exists h \exists \sigma : \begin{array}{l}
(h \in \mathcal{M}_c^{l_c}) \\
(\sigma \text{ a substitution of } L_c)
\end{array} \\
(\text{ground } C \sigma) \\
(\phi_c(C \sigma) = \text{false}) \end{array} \).

p2. Assume the antecedent of the theorem true, and let \( \sigma \) be as in the consequent of p1.

p3. By p2, p1, and Lemma 4.2--2 there exists an h' such that

\( h' \in \mathcal{M}_c^{ln} \)

\( (\phi_c'(T(C \sigma)) = \text{false}) \).

p4. From definitions 4.1.3, 2.4.6, p2, and general properties of substitutions, we have

\( T(C \sigma) = (T(C)) (T(\sigma)) \)

\( T(\sigma) \text{ a substitution of } L_m \)

\( T(\sigma) \text{ grounds } T(C) \).

p5. \( (\phi_c'(T(C)) = \text{false}, \text{ by p4, p3 and definition 2.4.8.3.} \)

p6. \( \bar{\theta}_c^{ln}(T(C)) = f, \text{ from definition 4.1.10, p3, and p5.} \)

Q.E.D.
The next lemma is not used explicitly in later theorems but is included for expository purposes. In Example III, section 4.3, it will be shown that the lemma does not hold with the word "ground" removed.

**Lemma 4.2 -- 4**

Let $C$ be a ground set of literals in $L_{c}$. Then

$$(S_{\Theta}(C) = t) \implies (S_{\Theta}(T(C)) = t).$$

**Proof:** Assume the antecedent of the lemma true.

1. Let $h'$ be any element of $M_{X}^{b m}$.

2. By definition 4.1.8 there exists an $h''$ such that $h'' \in M_{X}^{b m}$ and $h'' \subseteq h'$.

3. By definition 4.1.9 we can choose an $h$ such that

$$(h \in M_{X}^{b}) \text{ and } (T(h) = h'').$$

4. Then by definition 4.1.10 and the truth of the lemma antecedent we have $C \cap h \neq \emptyset$.

5. $T(C) \cap T(h) = T(C) \cap h'' \neq \emptyset$, from definition 4.1.3, and p4 and p3.

6. $T(C) \cap h' \neq \emptyset$, from p2.

7. $\phi_{\Theta}(T(C)) = \text{true}$, from definition 2.4.8.2.

8. From p7, p1, and definition 4.1.10, $S_{\Theta}(T(C)) = t$.

Q.E.D.
In the next theorem we first confront the second problem of using interpretations in resolution that Kowalski and Hayes pointed out, namely that of effective computability of truth values. The full solution to this problem will not be obtained until a later section, but here we must take the first step. This first step is reflected in two related features of the next theorem:

1. The theorem does not hold if "\(\longrightarrow\)" is replaced by "iff".

2. The property "satisfiable" is to have its usual meaning, i.e. a set of sentences, \(S\), is satisfiable iff there exists a relational structure, \(R\), such that each sentence in \(S\) is true in \(R\).

Intuitively one would want to have "satisfiable" replaced by a property like "true in at least one member of \(m^t\)", in which case the arrow could then be replaced by "iff". However, while such a theorem is intuitive, it is the unintuitive Theorem 4.2--5 which is the needed first step in circumventing the effective evaluation problem of interpretations. (N.B. We will write "\(K \cup \Sigma\)" instead of the more correct \("\{K\} \cup \Sigma\)" in all that follows.)

**Theorem 4.2 -- 5**

Let \(K\) be a finite set of literals in \(L_m\). Then

\[
(\theta^l_f(K) - f) \longrightarrow \text{(satisfiable \((\neg K) \cup \Sigma\))}
\]
proof: Let $K = \{k_1, k_2, \ldots, k_m\}$, and let $v_1, v_2, \ldots, v_j$
be all of the variable symbols in elements of $K$.

p1. Assume $\Theta^\text{bn}_\neg(K) = f$.

p2. There exists, by definitions 4.1.10 and 2.4.8.3, an $h$
and $\varphi$ such that:

(h $\in M^\text{bn}_\neg$)

($\varphi$ a substitution of $L_m$)

($\varphi$ grounds $K$)

($k_1 \varphi, k_2 \varphi, \ldots, k_m \varphi$ are all false ground
literals in $h$).

p3. From p2, the sentence

$$Q = \neg(k_1 \varphi) \land \neg(k_2 \varphi) \land \ldots \land \neg(k_m \varphi)$$

is a true sentence in $h$, i.e. $\varphi_A(Q) = \text{true}$.

p4. $\varphi_A(\exists v_1 \exists v_2 \ldots \exists v_j : \neg k_1 \land \neg k_2 \land \ldots \land \neg k_m) = \text{true}$, 
from p3 and definition 2.4.8.6.

p5. $\varphi_A(\neg K) = \text{true}$, from p4.

p6. From p2 and definition 4.1.6, $\varphi_A(\Xi) = \text{true}$.

p7. $\varphi_A((\neg K) \cup \Xi) = \text{true}$, by p5 and p6.

p8. (satisfiable ((\neg K) \cup \Xi)), from p7.

Q.E.D.

This last theorem is the crucial one for connecting the
model scheme functions ($\Theta$-functions) to a computable
property, or a partially computable property in some cases.
However, notice that the $\Theta$-functions are defined only for
sets of literals. We are next going to define two sound
semantic functions in terms of these $\Theta$-functions, and these
definitions will only explicitly define the semantic functions over sets of literals. We will most often refer to these partial functions as semantic functions, leaving the word "partial" as implicit.

4.2.1 A partial semantic function, $\Psi$, for a language $L$, is said to be a sound semantic function for $L$ iff

$\exists h$ (h a Herbrand interpretation for $L$)

$(\forall s: (s$ a sentence in $L)) ---->

[ (h(s) = false) ----> ( (\Psi(s) = false)

$\lor (\Psi$ not defined on $s$ ) ] ) ]$.

The last theorem of this section will show how any sound partial semantic function, which is defined only for sets of literals in $L$, can be extended to a sound semantic function for all sentences of $L$. First, however, we define some sound semantic functions for sets of literals.

Theorem 4.2 -- 6

Let $K$ range over sets of literals in $L_c$. Let

$$\Psi^c(K) = \begin{cases} 
\text{false} & \text{if } \Theta^c_{\nu}(K) = f \\
\text{true} & \text{if } \Theta^c_{\nu}(K) = t 
\end{cases}$$

Then $\Psi^c$ is a sound semantic function for $L_c$. 
proof: Since \( \mathcal{M}_{\mathcal{L}}^{bc} = \{ \} \), we can arbitrarily choose some \( h \) such that \( h \in \mathcal{M}_{\mathcal{L}}^{bc} \). Then \( \phi_h(K) = \text{false} \implies \Theta_{\mathcal{L}}^{bc}(K) = f \), by definition 4.1.10, and \( \phi_h(K) = \text{false} \implies \phi_h(K) = \text{false} \) by the theorem definition of \( \psi^c \). But \( \phi_h \) is sound and \( \psi^c \) is false permissive with respect to \( \phi_h \), thus \( \psi^c \) is sound.

Q.E.D.

Theorem 4.2 -- 7

Let \( K \) range over sets of literals in \( \mathcal{L}_c \). Let

\[
\psi^m(K) = \begin{cases} 
\text{false} & \text{if } \Theta_{\mathcal{L}}^{bm}(\mathcal{T}(K)) = f \\
\text{true} & \text{if } \Theta_{\mathcal{L}}^{bm}(\mathcal{T}(K)) = t
\end{cases}
\]

Thus \( \psi^m \) is a sound semantic function for \( \mathcal{L}_c \).

proof:

p1. \( \psi^c \) of Theorem 4.2-6 is a sound semantic function for \( \mathcal{L}_c \).

p2. \( \psi^c(K) = \text{false} \implies \Theta_{\mathcal{L}}^{bc}(K) = f \), by Theorem 4.2-6.

p3. \( \Theta_{\mathcal{L}}^{bc}(K) = f \implies \Theta_{\mathcal{L}}^{bm}(\mathcal{T}(K)) = f \), by Lemma 4.2-3.

p4. \( \psi^c(K) = \text{false} \implies \psi^m(K) = \text{false} \), from p2, p3, and the theorem definition of \( \psi^m \).

p5. \( \psi^m \) is false permissive with respect to \( \psi^c \), and \( \psi^c \) is sound, therefore \( \psi^m \) is sound, by p4, p1, and section 2.4.10.

Q.E.D.
Theorem 4.2 -- 8

Let K range over finite sets of literals in \( L_c \). Let

\[
\psi^\xi(K) = \begin{cases} 
\text{false} & \text{if (satisfiable } (T(\neg K) \cup \Sigma)) \\
\text{true} & \text{if (unsatisfiable } (T(\neg K) \cup \Sigma))
\end{cases}
\]

Then \( \psi^\xi \) is a sound semantic function for \( L_c \).

**proof:**

p1. \( \psi^m \) of Theorem 4.2-7 is a sound semantic function.

p2. \( \psi^m(K) = \text{false} \rightarrow \Theta^m_\Sigma(T(K)) = f \), by the definition of \( \psi^m \) in Theorem 4.2-7.

p3. \( \Theta^m_\Sigma(T(K)) = f \rightarrow (\text{satisfiable } (\neg T(K) \cup \Sigma)) \), by Theorem 4.2-5.

p4. From definition 4.1.3, we have \( \neg T(K) = T(\neg K) \).

p5. \( \psi^m(K) = \text{false} \rightarrow (\text{satisfiable } (T(\neg K) \cup \Sigma)) \), from p2, p3 and p4.

p6. \( \psi^m(K) = \text{false} \rightarrow \psi^\xi(K) = \text{false} \), from p5 and the theorem definition of \( \psi^\xi \).

p7. \( \psi^\xi \) is false permissive with respect to a sound semantic function, therefore \( \psi^\xi \) is sound, from p1 and p6.

Q.E.D.
\( \psi_\xi \) is a sound semantic function which, assuming a decision procedure were available for satisfiability testing for \( \lambda \xi \), could actually be implemented. If the resolution strategy HLR - t.v.d. were to be used, then \( \psi_\xi \) would be sufficient as it is, i.e. defined only for sets of literals in \( \lambda \xi \). If HLR is being used however, \( \psi_\xi \) would have to be extended so as to also assign truth values to the \( \alpha \) - and \( \emptyset \) - statements of clauses. In order for HLR to be complete, the only requirement placed on the semantic function is that it be sound. Because of this the next theorem is concerned only with defining a sound total semantic function based on a sound partial semantic function, and not with the issue of whether the total function conforms to the usual notions of truth. In fact the extension stated in the theorem does conform to intuitive notions of truth values for \( \alpha \) - and \( \emptyset \) - statements. This has relevance for efficiency when the total semantic function is used in an actual HLR search, but does not influence completeness.
Theorem 4.2 -- 9

Let \( \Psi \) be a sound partial semantic function for \( L \), defined only for (all) sets of literals in \( L \). For \( H \) an arbitrary clause such that \( L \) a language of \( H \), where

\[ H = \{ s_1, \ldots, s_i, \{ r_1, \ldots, r_f \} \} \]

let \( \Psi \) be extended as follows:

\[ \Psi(\alpha(H)) = \Psi(\{r_1, \ldots, r_f, s_1, \ldots, s_i\}) \]

\[ \Psi(\beta(H)) = \text{true} \]

iff

\[ \{(\Psi(\{r_1, \ldots, r_f, \neg s_1\}) = \text{true}\}
\]

\[ \land (\Psi(\{r_1, \ldots, r_f, \neg s_2\}) = \text{true}\}
\]

\[ \vdots \]

\[ \land (\Psi(\{r_1, \ldots, r_f, \neg s_i\}) = \text{true}\} \]

\[ \Psi(x) = \text{false} \] if \( x \) is a sentence of \( L \) which is neither a set of literals nor the \( \alpha \)- or \( \beta \)-statement of any clause in \( L \).

Then such an extended \( \Psi \) is a sound total semantic function for \( L \).
proof: Clearly the extended $\Psi$ is a total semantic function for $L$, and it remains to show that it is sound.

p1. The extended $\Psi$ applied to sets of literals is sound, by the theorem hypothesis.

p2. There exists an $h$ such that

\[
\forall K \ ( (K \text{ a set of literals in } L) \implies (\phi_h(K) \implies \neg \Psi(K)) \implies \Psi(K) = \text{false} ) \),
\]

by p1.

p3. For $H$ as in the theorem statement,

\[
\phi_h(\emptyset(\{\})) = \phi_h(\{r_1, r_2, \ldots, r_k, s_1, \ldots, s_{\ell}\})
\]

from definitions 2.4.13, 2.4.8 and the usual meaning of logical symbols.

p4. If $\phi_h(\emptyset(\{\})) = \text{false}$, then

\[
\phi_h(\{r_1, \ldots, r_k, s_1, \ldots, s_{\ell}\}) = \text{false},
\]

and

\[
\Psi(\{r_1, \ldots, r_k, s_1, \ldots, s_{\ell}\}) = \text{false},
\]

and thus $\Psi(\emptyset(\{\})) = \text{false}$, from p3, p2 and c2.

p5. $\phi_h(\emptyset(\{\})) = \phi_h(\forall x:\ (\neg r_1 \land \neg r_2 \land \cdots \land \neg r_k) \implies (-s_1 \land \cdots \land \neg s_{\ell})))$

from 2.4.13.

p6. $\phi_h(\emptyset(\{\})) = \phi_h(\forall x:\ (r_1 \lor r_2 \lor \cdots \lor r_k \lor \neg s_1) \land (r_1 \lor r_2 \lor \cdots \lor r_k \lor \neg s_2) \land \cdots \land (r_1 \lor r_2 \lor \cdots \lor r_k \lor \neg s_{\ell}))$

from p5, and the usual properties of logical connectives and quantifiers.
p7. \( \varphi_A(\emptyset(\Lambda)) = \text{true} \) iff
\[
\land (\varphi_A(\{r_1, \ldots, r_i, \neg s_i\}) = \text{true}) \\
\land (\varphi_A(\{r_1, \ldots, r_i, \neg s_i\}) = \text{true}) \\
\vdots \\
\land (\varphi_A(\{r_1, \ldots, r_i, \neg s_i\}) = \text{true})
\]
by the default interpretation of a set of literals as a universally closed disjunction, and the meaning of "\(\land\)".

p8. Assume \( \varphi_A(\emptyset(\Lambda)) = \text{false} \), then there exists an \( n \), 1 \( \leq n \leq i \), such that \( \varphi_A(\{r_1, \ldots, r_i, \neg s_n\}) = \text{false} \).

p9. Then \( \gamma(\{r_1, \ldots, r_i, \neg s_n\}) = \text{false} \), by p2.

p10. Thus \( \gamma(\emptyset(\Lambda)) = \text{false} \), from p8, p9 and c3.

p11. \( \gamma \) (as a total function) is false permissive wrt. \( \varphi_A \),
by p2, p4, p10 and c4.

p12. \( \gamma \) is a sound total semantic function for \( L \), by p2
and p11.

Q.E.D.

Note that an actual HLR search will only utilize \( \gamma \) for evaluating sets of literals (feasibility tests) and for \( \alpha \)- and \( \xi \)-statements of feasible clauses (for t.v.d. evaluations). Thus the part of the theorem statement that says "\( \gamma(x) = \text{false} \) if \( \{x \text{ is a } \ldots \} \)" is included only to be able to easily assert the existence of a total function as the theorem consequence, and means nothing in practical situations for HLR.
Because of Theorem 4.2-9 our sound semantic functions on sets of literals have implicit in them the extension to sound total semantic functions, and we need not explicitly define our sound semantic functions to be total.

Taking Theorems 4.2-8 and 4.2-9 together we now have the definition of a sound (total) semantic function which could be used in an HLR search process, provided we had a decision procedure for satisfiability testing of sets of sentences of the form

\[ T^{(K)} \cup \Sigma \]

where \( K \) is a set of literals in \( L_e \). Of course in general there will be no decision procedure possible, but in some practical situations a decision procedure would exist. When a decision procedure exists and is not too computationally costly to actually use in an implementation, then we have circumvented the second difficulty pointed out by Kowalski and Hayes, listed at the beginning of this chapter. In section 4.4 we will show how this difficulty may be overcome in those cases where there is no suitable decision procedure.

Notice that we have already overcome the first problem listed at the beginning of this chapter, namely that of having to be able to explicitly specify the truth values of literals containing Skolem functions. To see this, consider Theorem 4.2-8. Suppose \( L_e \) contained an \( n \)-place function, "\( f \)", which is a Skolem function, and therefore occurs in the
input set of clauses. Thus, for at least some of the sets of literals, \( K \), that \( \psi^\Sigma \) will have to evaluate, \( K \) will contain a literal in which "f" occurs. We have the implicit assumption that \( L_c \), \( L_m \), \( T \) and \( \Sigma \) are components of a model scheme, so that there must be an \( n \)-place function symbol, "g" (say), in \( L_m \), such that

\[
T(f) = g.
\]

However, there is no requirement that \( \Sigma \) actually contain any occurrences of the function "g", or, if it does, that it in any sense totally defines "g". Thus, in choosing \( \Sigma \), one is free to include as much or as little as desired about "f", by including sentences in \( \Sigma \) that constrain "g" in whatever way one wishes to (subject only to the constraint that \( \Sigma \) must be satisfiable).

It is important to realize that given a specific \( L_c \), \( L_m \) and \( T \), that there is still enormous freedom in deciding what function \( \psi^\Sigma \) is, by the freedom to choose \( \Sigma \).

In the next section we consider an example of a model structure such that \( \psi^\Sigma \) is effectively computable, and in section 4.4 we will consider what is to be done in situations where \( \psi^\Sigma \) is not a practical function.
4.3 Example III - A Simple Model Structure.

In this section we illustrate a very simple model structure in which $L_c$ has two constants, no other functions, and just the usual equality relation. The reason for treating such a simple example is to illustrate a model structure by actually explicitly giving the model schemes involved, and to be able to easily compute the $\theta$-functions. In section 4.5 a more sophisticated example is treated.

Consider the model structure

$$M = < L_c , L_m , T , \Sigma >$$

with

$$L_c : \text{desc}(L_c) = < (\{0, 0\}, \{c_1, c_2\}) >$$

$$L_m : \text{desc}(L_m) = < (\{R(0, 0)\}, \{k_1, k_2, k_3\}) >$$

and $T$ is induced by the amicability function

$$\Psi = \{ <c_1, k_1>, <c_1, k_2>, <c_1, k_3>, <\leq, R> \}.$$

We take the scheme defining set of statements, $\Sigma$, to be the following 7 clauses in $L_m$:

1. $\neg R(k_1, k_2)$;
2. $\neg R(k_1, k_3)$;
3. $\neg R(k_2, k_3)$;
4. $R(x, x)$;
5. $\neg R(x, y), R(y, x)$;
6. $\neg R(x, y), \neg R(y, z), R(x, z)$;
7. $R(x, k_1), R(x, k_2), R(x, k_3)$;


It is easily seen that $T$ translates $L_c$ to $L_m$, and
that there is a Herbrand interpretation for $L_m$,

$$h_\Sigma = \{ R(k_1, k_1), R(k_2, k_2), R(k_3, k_3), \neg R(k_4, k_4), \neg R(k_5, k_5), \neg R(k_6, k_6), \neg R(k_7, k_7), \neg R(k_8, k_8) \}$$

which satisfies $\Sigma$ (in fact, $h_\Sigma$ is the only Herbrand interpretation for $L_m$ that satisfies $\Sigma$). Therefore $M$ meets all of the conditions required for it to be a model structure.

For this $M$, definitions 4.1.6 to 4.1.9 give

$$\mathcal{M}_\Sigma^{L_m} = \{ h_\Sigma \}, \quad \text{and therefore } \phi^{L_m}_{\Sigma} \iff \phi^{L_c}_{\Sigma} = \text{false}$$

$$\text{desc}(L_c \setminus L_m) = \langle \{ R(-, -) \}, \{ k_1, k_2 \} \rangle$$

$$\mathcal{M}_\Sigma^{L_c \setminus L_m} = \{ \neg R(k_1, k_1), R(k_2, k_2), \neg R(k_3, k_3), \neg R(k_4, k_4), \neg R(k_5, k_5) \}$$

$$\mathcal{M}_\Sigma^{L_c} = \{ \neg c_1, c_1, \neg c_2, c_2, \neg c_3, c_3 \}. \quad \text{Thus each of the three model schemes induced by } M \text{ is a singleton set.}$$

Let us now look at the evaluation process for the $\Theta$-evaluation functions. Consider the normal clause (set of literals)

$$Q_i = \{ \neg (x, c_1), = (x, c_2) \}.$$ Then $\phi^{L_c}_{\Sigma}(Q_i) = t$, since the only ground terms that can be substituted for $x$ are $c_1$ and $c_2$, and whichever is chosen, the resulting set of literals is true in the only interpretation in $\mathcal{M}_\Sigma^{L_c}$. 
Similarly
\[ \Theta^L_\Sigma (T(Q_1)) = \Theta^L_\Sigma (\{ R(x, k_1), R(x, k_2) \}) = t, \]
since the only ground terms available are \( k_1 \) and \( k_2 \) in the Herbrand universe of \( L_c \setminus L_m \), and substituting either of these for \( x \) leaves one of the literals true in the only interpretation in \( \mathcal{M}_\Sigma^L \).

However, \( \Theta^L_\Sigma (T(Q_1)) = f \), since now \( \varphi = \{ k_2 / x \} \) will ground \( T(Q_1) \) to
\[ \{ R(k_2, k_1), R(k_3, k_2) \} \]
which is false in the only interpretation in \( \mathcal{M}_\Sigma^L \). This clearly is a result of \( L_m \) having a larger Herbrand universe than \( L_c \setminus L_m \). This illustrates how Lemma 4.2-4 cannot have the word "ground" removed and still hold in general.

We now show the evaluation of \( \psi^\Sigma(Q_1) \), for the \( \psi^\Sigma \) of Theorem 4.2-8. We form \( T(\neg Q_1) \cup \Sigma \) and see if it is satisfiable.

\[ T(\neg Q_1) = T(\exists x: (\neg = (x, c_1) \land \neg = (x, c_2)) \land \exists x: (\neg R(x, k_1) \land \neg R(x, k_2)) \]

Now \( T(\neg Q_1) \cup \Sigma \) is satisfiable, as can be seen by letting \( x \) in \( T(\neg Q_1) \) be \( k_9 \), and noting that
\[ \phi^L_\Sigma (\neg R(k_3, k_1) \land \neg R(k_2, k_2)) = \text{true} \]
(and, of course, \( \phi^L_\Sigma (\Sigma) = \text{true} \), by the definition of \( h_\Sigma \)). Thus \( \psi^\Sigma(Q_1) = \text{false} \).
Notice that what was just done to evaluate $\psi^\xi(Q_1)$ is to show that in the relational structure induced by $h_\xi$, that $T(\neg Q_1) \cup \Sigma$ was true, and therefore is satisfiable, so that Theorem 4.2-8 gives $\psi^\xi(Q_1) = \text{false}$. In general this is not an approach that will work however, since the relation "satisfiable" of Theorem 4.2-8 is not meant to be restricted to mean "satisfiable over relational structures induced by Herbrand interpretations in $M^L_\Sigma$". This occurs because Theorem 4.2-8 is based on Theorem 4.2-5. Thus even if $T(\neg Q_1)$ is false in every Herbrand interpretation in $M^L_\Sigma$, we cannot assert that $\psi^\xi(Q_1)$ is true, since there may exist other relational structures in which $\Sigma$ and $T(\neg Q_1)$ are both true, and therefore $T(\neg Q_1) \cup \Sigma$ is satisfiable, and $\psi^\xi(Q_1) = \text{false}$. Therefore what we really need is an evaluation mechanism for detecting unsatisfiability, and we now show the evaluation of $\psi^\xi(Q_1)$ using this approach.

A natural approach to doing this, since we are considering models for resolution procedures, is to take $T(\neg Q_1) \cup \Sigma$ and convert it to Skolem conjunctive normal form, and test the resulting clauses for unsatisfiability. Doing this gives us the 7 clauses of $\Sigma$ along with

8. $\neg R(a,k_1)$;
9. $\neg R(a,k_2)$;

where "a" is a new Skolem constant. Since the Herbrand universe of this set is finite, there are decision procedures for satisfiability testing (one way is to produce
all the possible Herbrand interpretations and see if they
all fail to satisfy the set. Here we verify the
satisfiability of these 9 clauses by giving an
interpretation in which all of the clauses are true:

\[ h = \{ R(a, a), \neg R(a, k_1), \neg R(a, k_2), \neg R(k_1, a), \neg R(k_2, a), R(a, k_3), R(k_2, a) \} \cup b_\Xi. \]

Therefore, by Theorem 4.2-8, \( \psi^S(Q) = \text{false}. \)

As another example consider the set of literals
\( Q_\alpha = \{ = (x, c_1), = (x, c_2), = (y, c_1), = (y, c_2), = (x, y) \}. \)

It is easy to see that
\[ \Theta_\delta^{\mu_\alpha}(Q_\alpha) = \top \]
\[ \Theta_\delta^{\mu_\alpha}(T(Q_\alpha)) = \top \]
and
\[ \Theta_\delta^{\mu_\alpha}(T(Q_\alpha)) = \top. \]

We show that \( \psi(\xi) = \text{true} \) by converting \( T(\neg Q_\alpha) \cup \Xi \) to
clause form and exhibiting a refutation. The clauses are
the 7 clauses of \( \Xi \), plus

8. \( \neg R(c, k_1); \)
9. \( \neg R(a, k_2); \)
10. \( \neg R(b, k_1); \)
11. \( \neg R(b, k_2); \)
12. \( \neg R(a, b); \)

where "a" and "b" are new Skolem constants.
The refutation for this set of clauses is:

7x8 = 13. R(a,k_2), R(a,k_3);
9x13 = 14. R(a,k_2);
7x10 = 15. R(b,k_2), R(b,k_3);
11x15 = 16. R(b,k_2);
5x16 = 17. R(k_3,b);
6x14 = 18. \neg R(k_3,z), R(a,z);
17x18 = 19. R(a,b);
12x19 = 20. \square.

Therefore \( \gamma^x(Q) = \text{true} \).

As can be seen from the \( \gamma^x \) computations for \( Q_1 \) and \( Q_2 \), the evaluation process is relatively simple for the \( \Sigma \) of this example. The next section deals with issues involved in more sophisticated models, where satisfiability testing presents computational difficulties because of either

1. the language, \( L_m \), being undecidable (or, more properly, \( \Sigma \) an undecidable theory in \( L_m \));

or

2. the available decision procedures are too inefficient for practical purposes.
4.4 Pragmatic Construction of Sound Semantic Functions.

In applying the material of the previous sections to an implementation of a theorem prover which utilizes a semantic function, several problems and inadequacies with the method become apparent. Some of these inadequacies will be discussed later, but three of them are listed here, as it is the purpose of this section to show how these may be overcome, at least partially. The first two are just a finer categorization of one of our original problems with interpretations, which is that of effective and efficient computability of truth values.

1. The satisfiability testing, on which a semantic function such as $\gamma^L$ of Theorem 4.2-8 is based, may be impossible to perform because of the undecidability of the class of sets of sentences that have to be tested.

2. The class of sets of sentences for which satisfiability is to be determined may be a decidable class, but the computation is too costly to perform for some sentences.

3. The semantic function is static in the sense of not being modified by information which becomes available during the search process.

Items 1 and 2 above are closely related. We deal first with a switch from $\gamma^L$ as the semantic function, to a semantic function which we call $\gamma^R$. $\gamma^R$ adequately deals with item 1, and in some specific model structures may
adequately deal with item 2. We then define a type of sound
semantic function, denoted by $\psi^c$, which for some model
structures can be superior to $\psi^R$ in handling item 2 above.

4.4.1 Suppose $\mathcal{R}$ is a first order refutation procedure
which accepts a set of sentences (e.g. clauses) in the
language $L$, searches for a refutation for some finite amount
of time, and then terminates, reporting either "success" or
"failure". Further suppose that $\mathcal{R}$ is sound, i.e. $\mathcal{R}$
always reports "failure" when given a satisfiable set of
sentences. Such a procedure is called a sound and
terminating refutation procedure for $L$. We write $\mathcal{R}(x)$ to
indicate the reporting value of $\mathcal{R}$ acting on the set of
sentences $x$.

Theorem 4.4 -- 1

Let $\mathcal{R}$ be a sound and terminating refutation procedure
for $L_m$, and define $\psi^\mathcal{R}$ by:

for any finite set of literals, $K$, in $L_c$,

$$\psi^\mathcal{R}(K) = \begin{cases} 
\text{false} & \text{if } \mathcal{R}(T(\neg K) \cup \Sigma) = \text{failure} \\
\text{true} & \text{if } \mathcal{R}(T(\neg K) \cup \Sigma) = \text{success} 
\end{cases}$$

Then $\psi^\mathcal{R}$ is a sound semantic function for $L_c$. 
proof: Immediate from the soundness of $\psi^\xi$ of Theorem 4.2-8 and the definition of $R$ and $\psi^R$, by which $\psi^R$ is false permissive with respect to $\psi^\xi$.

Q.E.D.

$\psi^R$ is a semantic function which can be computed, although for some model structures the cost may be high if we try to make the frequency of "extra" false answers low. However in some practical situations one may find that a particular model structure, which seems to be a good model for various reasons, has the property that for most sets of literals the satisfiability checking is easy, but that for a very few sets the satisfiability checking is extremely costly. The use of $\psi^R$ as the semantic function can, by suitably structuring $R$, bypass the very difficult evaluations.

For example, if $R$ is itself a resolution procedure, and $\mathcal{T}(\neg R) \cup \xi$ is put into clause form, one could choose any resolution strategy (complete or incomplete, but sound) which one suspected would rapidly refute most of the refutable clause sets, and couple it with a search cutoff after some fixed number of clauses. In some particularly simple models this may make it practical to actually use a general purpose theorem prover to perform model evaluations. The more likely situation however is that the model language $L_m$ and the defining set of statements $\xi$ will be much too complex to be dealt with by a general purpose theorem
proving procedure. In these cases special techniques which are specific for the model will be necessary.

We now address the question of constructing sound semantic functions from evaluation procedures which are domain specific. We use a procedure, \( \mathcal{P} \), to define a semantic function, \( \psi^{\mathcal{P}} \), as a partial semantic function defined explicitly for sets of literals, and assume the implicit extension to all sentences as given by Theorem 4.2-9. We are again concerned with stating the criteria by which we can assert the soundness of \( \psi^{\mathcal{P}} \).

Suppose \( \mathcal{P} \) is a procedure (e.g. a computer coded procedure) which terminates for all inputs which are sentences in the first order language \( L \), and returns either the value "inconsistent" or the value "unknown". Then \( \mathcal{P} \) is called a terminating inconsistency detector for \( L \). If \( s \) is a sentence of \( L \), then we denote by \( \mathcal{P}(s) \) the value returned by \( \mathcal{P} \) on termination when starting with \( s \) as input. Define \( J(\mathcal{P}, L, \Xi) \) to be the notational shorthand:

\[
J(\mathcal{P}, L, \Xi) \triangleq \forall s: \{ (s \text{ a sentence of } L) \quad \rightarrow \quad ((\mathcal{P}(s) \text{ inconsistent})
\quad \quad \rightarrow \quad (\text{unsatisfiable } (s \cup \Xi)) \}) \}.
\]

We say \( J(\mathcal{P}, L, \Xi) \) is the \( \Xi \)-justification statement for \( \mathcal{P} \) in \( L \).
Theorem 4.4 -- 2

Let $M < L\epsilon, L_m, \mathcal{T}, \Xi >$ be a model structure, $\rho$ be a terminating inconsistency detector for $L_m$, and suppose $J(\rho, L_m, \Xi)$ is true. Let $\psi^\rho$ be defined, for all finite sets of literals, $K$, in $L\epsilon$, by

$$
\psi^\rho(K) = \begin{cases} 
\text{false} & \text{if } \rho(\mathcal{T}(\neg K)) = \text{unknown} \\
\text{true} & \text{if } \rho(\mathcal{T}(\neg K)) = \text{inconsistent}.
\end{cases}
$$

Then $\psi^\rho$ is a sound semantic function for $L\epsilon$.

proof: If $\psi^\rho(K) = \text{true}$, then $\rho(\mathcal{T}(\neg K)) = \text{inconsistent}$. But then we have (unsatisfiable $\mathcal{T}(\neg K) \cup \Xi$) from the theorem hypothesis that $J(\rho, L_m, \Xi)$ holds. Then, from Theorem 4.2-8, $\psi^\xi(K) = \text{true}$. Thus $\psi^\rho(K) = \text{true}$ implies $\psi^\xi(K) = \text{true}$. Therefore $\psi^\rho$ is false permissive wrt. $\psi^\xi$, which is a sound semantic function for $L\epsilon$, and thus $\psi^\rho$ is a sound semantic function for $L\epsilon$.

Q.E.D.
Example IV will illustrate this use of model domain specific techniques for semantic function evaluations. First however we give a partial treatment of item 3 from the beginning of this section, as this will also be illustrated in Example IV.

When initially setting up a model structure for a set of clauses, in $L_c$, one is presented with a situation where it is usually unclear just what should be the exact character of $\mathcal{K}$ (assuming some $L_m$ and $\mathcal{T}$ have already been chosen). In particular there are generally sentences in $L_m$ for which the effects on the search space (in $L_c$) of including them in $\mathcal{K}$ is unknown. Suppose $s$ is a sentence in $L_m$ such that $s \cup \mathcal{K}$ and $\neg s \cup \mathcal{K}$ are both satisfiable. Then there are three possibilities: either $s$ or $\neg s$ could be added to $\mathcal{K}$, or neither could be added. It seems reasonable to assume that as the search space is developed in a search process, more information relevant to determining the structure of $\mathcal{K}$ will become available. This raises the question of the completeness of the search when $\mathcal{K}$ varies during the search. We address this question here by giving a description of one type of model change which does conserve completeness.
4.4.2 Let \( I \) be a finite or infinite set of integers, and let

\[
\Sigma^i = \{ \Sigma^i \mid i \in I \}
\]

where each \( \Sigma^i \) is a satisfiable set of sentences in \( L_m \). Then \( \Sigma^i \) is called an overlapping class of schemes (with index set \( I \)) iff

\[
\bigcap_{i \in I} m_{\Sigma^i}^m \neq \{\}.
\]

4.4.3 An \textbf{I-indexed semantic function} for \( L \) is a function, total on \( \{ \text{sets of literals in } L \} \times I \) and into \( \{ \text{true, false} \} \).

An I-indexed semantic function, \( \psi^I \), for \( L \) is said to be sound iff

\[
\exists h: \{ (h \text{ a Herbrand interpretation for } L) \}
\]

\[
(\forall i \in I: (K \text{ a finite set of literals of } L) \\
(\forall i \in I) \left( \phi^h(K) = \text{false} \rightarrow \psi^I(K,i) = \text{false} \right) )
\].

4.4.4 For \( K \) a set of literals in \( L_c \), \( i \in I \), \( \mathcal{R} \) a sound and terminating refutation procedure for \( L_m \), and \( \Sigma^I \) an overlapping class of schemes with index set \( I \), we define the I-indexed semantic function for \( L_c \), \( \Sigma^I \psi^I \):

\[
\Sigma^I \psi^I(K,i) = \begin{cases} 
\text{false} & \text{if } \mathcal{R}(\langle I^i \rangle K \cup \Sigma^I) = \text{failure} \\
\text{true} & \text{if } \mathcal{R}(\langle I^i \rangle K \cup \Sigma^I) = \text{success}
\end{cases}
\]
Lemma 4.4 -- 3

For $\mathcal{I}$ as defined above, it is the case that

$$\exists h: \{ (h \text{ a Herbrand interpretation for } L_\mathcal{I})$$

$$\forall i \forall K: \{ (K \text{ a finite set of literals in } L_\mathcal{I})$$

$$(i \in I)$$

$$\rightarrow (\phi \gamma (K) = \text{false} \rightarrow \mathcal{I} \gamma (K,i) = \text{false}) \}$$

That is, $\mathcal{I} \gamma$ is a sound $I$-indexed semantic function.

proof: Since $\mathcal{I}$ is an overlapping class, there exists an $h^*$ such that $h^* \in \mathcal{M}^{\uparrow \mathcal{I}}_\mathcal{I}$, for all $i \in I$. Let $h$ be a Herbrand interpretation for $L_\mathcal{I}$ such that $\mathcal{T}(h) \subseteq h^*$. We now show that this $h$ can serve as the $h$ of the lemma statement.

Assume $\phi \gamma (K) = \text{false}$.

p1. Then for any $i \in I$,

$$\mathcal{E}^{\uparrow \mathcal{I}}_\mathcal{I}(K) = f,$$

by definition 4.1.10.

p2. By p1 and Lemma 4.2-3,

$$\mathcal{E}^{\uparrow \mathcal{I}}_\mathcal{I}(\mathcal{T}(K)) = f.$$

p3. By p2 and Theorem 4.2-5,

$$(\text{satisfiable } (\mathcal{T}(\neg K) \cup \mathcal{L}_\mathcal{I})).$$

p4. $\mathcal{R}(\mathcal{T}(\neg K) \cup \mathcal{L}_\mathcal{I}) = \text{failure by } p3 \text{ (and because } \mathcal{R} \text{ is sound and terminating).}$

p5. $\mathcal{I} \gamma (K,i) = \text{false by } p4 \text{ and the definition of } \mathcal{I} \gamma$.

Q.E.D.
4.4.5 Let I be the set of positive integers. An HLR search process is said to be using an I-indexed semantic function, \( \hat{\mathcal{L}} \), if for all \( j \), the \( j \)-th set of literals, \( K_j \), to be evaluated during the search has its truth value taken to be \( \hat{\mathcal{L}}(K_j,j) \).

The next theorem shows that HLR is complete when using \( \hat{\mathcal{L}} \) in a breadth first search process. To be completely defined it is necessary to specify how and when t.v.d. and feasibility evaluations are done since the resulting search space content depends upon the integers used as the second argument to \( \hat{\mathcal{L}} \). The reader should be able to see, from the proof of the next theorem, that refutation completeness is not dependent upon the exact way this is done, but that the search space overall will depend upon how this is done. Virtually any intuitively plausible rule for when and how feasibility and t.v.d. evaluations are done will be complete. However it is assumed that \( \alpha \)- and \( \theta \)-statements are evaluated by \( \hat{\mathcal{L}} \) by applying \( \hat{\mathcal{L}} \) to sets of literals as in Theorem 4.2-9.
Theorem 4.4 -- 4

Let £ have a finite number of elements. Let I be the set of positive integers and let $\mathcal{F}$ be as defined in section 4.4.4.

If [ (L a language for £) (HL-proper £) (unconstrained £) (unsatisfiable £) ]

then a breadth first HLR search process using $\mathcal{F}$, and starting from £, will generate a null clause after some finite number of other clauses have been generated.

proof: Let h be a Herbrand interpretation whose existence is asserted by Lemma 4.4-3. Then by the Herbrand model refutation completeness of HLR (Theorem 3.1-2), there exists some HLR refutation, $R_h$ (using $\phi_h$), of £. Let the depth of $R_h$ be D. Let Z be the HLR search space expansion using $\mathcal{F}$ out to and including all level D clauses, i.e. Z is a sequence of deductions, starting with level 0 deductions (i.e. input clauses) and followed by level 1 deductions, etc. By considering Z to be a sequence of deductions we intend that distinct deductions having the same root clause are all retained in Z. Notice that all of the leaves in $R_h$ are in the search space expansion of £ using $\mathcal{F}$, since they are in £ and are unconstrained, and are therefore $\mathcal{F}$-feasible (however the t.o.d. values may be different than in $R_h$). Also notice that Z is of finite length.
Assume $R_A$ (for all possible re-assignments of t.v.d. values on clauses in $R_A$) is not part of the search space expansion, $Z$. Then there is a triple of clauses $H', H^2, H^3$ such that in $R_A$

$$H' \times H^2 \xrightarrow{\phi_H^{ME}} H^3$$

and such that the deductions of $H'$, $H^2$ are both in $Z$, but the deduction of $H^3$ (with the deduction of $H'$ and the deduction of $H^2$ the parent deductions of $H^3$) is not in $Z$. Notice that the depth of $H^i < d$, for $i = 1, 2$. Let $N = \max(\text{depth of } H', \text{depth of } H^2)$. But the meaning of a breadth first search is that in forming clauses at level $N + 1$, all possible clause pairs, with each clause at level $N$ or below, are considered as possible parents (both ways, i.e. each resolved against the other). Thus $H'$ and $H^2$ are chosen as parents. We now argue that since $H'$ and $H^2$ resolve together in $R_A$, that the t.v.d. and feasibility conditions are correct (in $Z$) so that

$$H' \times H^2 \xrightarrow{\phi_H^{ME}} H^3$$

independently of what integers were used for the second argument of $\phi_I^\psi$ for

1. the t.v.d. evaluations of $H'$ and $H^2$ when they were produced;
2. the t.v.d. and feasibility evaluations for the factors of $H'$ and $H^2$, and the binary inference step between the factors (at the time that $H'$ and $H^2$ are chosen as potential parents).
The argument is virtually identical to that for the last part of the proof of Theorem 3.1-1 (False Permissive Completeness of HLR), except that now instead of calling on Lemma 3.4-1 (A t.v.d. Conservation Property), we use instead Lemma 4.4-3. Given

\[ H' \times H^2 \vdash_{\phi^R} \sim H^5 \]

in \( R^\phi \), we can assert that t.v.d.\((H', \phi^R) \not\models T \). This means that \( \phi^R(\sim(H')) = \text{false} \). Thus, by Lemma 4.4-3 and Theorem 4.2-9,

\[ \forall i : \psi(\sim(H'), i) = \text{false} \]

Therefore t.v.d.\((H', \psi') \not\models T \), independently of the integers used for the second argument of \( \psi' \) when the t.v.d. evaluation of \( H' \) was done. A similar argument establishes that the t.v.d. of \( H^2 \) is not \( F \), so item 1 above has been verified. By similar arguments item 2 can be verified. Thus \( Z \) must include \( H^3 \) as a level \( N+1 \) deduction, for some \( N < 0 \), and the parents of \( H^3 \) are \( H' \) and \( H^2 \). But the assumption above that \( R^\phi \) (with t.v.d. values possibly changed) was not part of \( Z \) led immediately to the existence of \( H^3 \) as a clause not deduced from \( H' \) and \( H^5 \). Thus we have a contradiction, and it must be the case that for some re-assignment of t.v.d.'s that \( R^\phi \) is in \( Z \). Thus the breadth first search using \( \psi' \) will generate a null clause.

Q.E.D.
Clearly the above proof depends upon $\Psi$ only through Lemma 4.4-3, and thus $\Psi$ may be replaced in the Theorem statement by any sound $I$-indexed semantic function for $L_0$, where $I$ is the set of positive integers.

4.4.6 If $\xi$ is an overlapping class of schemes with index set $I$, and

$$\forall i \forall j: (i \in I) (j \in I) (i > j)$$

$$\longrightarrow (\xi_i \models \xi_j)$$

then $\xi$ is called a strengthening sequence (over $I$). When $I$ is the set of positive integers, then a strengthening sequence, $\xi$, will be written as an actual list:

$$\xi = \xi_1, \xi_2, \ldots$$

Example IV will illustrate a special case of a strengthening sequence, where

$$\xi_i \subseteq \xi_j$$

for $i > j$. 
4.4.7 Let $\mathcal{I}$ be a strengthening sequence (over the index set of all positive integers) such that $\mathcal{I}_j \subseteq \mathcal{I}_i$ for $i > j$.

Then there exists a $\mathcal{I}$ such that

$$\forall i: (i \in I) \implies \mathcal{I} \subseteq \mathcal{I}_i.$$

For each $i \in I$, we define $\mathcal{I}_i = \mathcal{I}_i - \mathcal{I}$, and thus

$$\mathcal{I}_i = \mathcal{I}_i \cup \mathcal{I}.$$

We call each $\mathcal{I}_i$ a context set, and the sequence

$$\mathcal{I}_1, \mathcal{I}_2, \ldots$$

a strengthening sequence of context sets, which we will denote by $\mathcal{Q}$. Suppose $M = < L_\mathcal{I}, L_m, T, \mathcal{I} >$ is a model structure, and that $\mathcal{Q}_\mathcal{I}$ is a terminating inconsistency detector for $L_m$ such that $J(\mathcal{Q}_\mathcal{I}, L_m, \mathcal{I})$ is true. Then define

$$\mathcal{Q}_\mathcal{I}(K, i) = \begin{cases} 
\text{false} & \text{if } \mathcal{Q}_\mathcal{I}(T(\neg K) \cup \mathcal{I}_i) = \text{unknown} \\
\text{true} & \text{if } \mathcal{Q}_\mathcal{I}(T(\neg K) \cup \mathcal{I}_i) = \text{inconsistent}
\end{cases}$$

In practice $\mathcal{I}$ will represent some satisfiable theory in $L_m$ which will justify the procedure $\mathcal{Q}_\mathcal{I}$, while the sequence $\mathcal{Q}$ will represent the increasing amount of information beyond that in the theory $\mathcal{I}$ which is to be used by $\mathcal{Q}_\mathcal{I}$ in the evaluation process. One can think of $\mathcal{I}$ as the part of the model structure which is procedurally defined, and each element in the sequence $\mathcal{Q}$ as the information in (the current state of) the model structure which remains as declarative information.
We assert that $\mathcal{C}_\mathcal{P}^{\mathcal{E}}$ is a sound semantic function, and thus can take the place of $\mathcal{C}$ in Theorem 4.4-4 and the theorem still holds. The proof of this uses arguments quite similar to those already used in this chapter. The overall plan of such a proof is as follows:

1. Show for each $K$ a sentence of $L_\mathcal{C}$, and $i \in I$, that
   
   \[ \mathcal{C}_\mathcal{P}(K,i) = \text{false} \rightarrow \mathcal{C}_\mathcal{P}^{\mathcal{E}}(K,i) = \text{false} \];

2. state and prove (using 1 above) the analogue of Lemma 4.4-3, where $\mathcal{C}$ is replaced by $\mathcal{C}_\mathcal{P}$;

3. the proof of Theorem 4.4-4 is correct as it is with " $\mathcal{C}$ " replaced by " $\mathcal{C}_\mathcal{P}^{\mathcal{E}}$ ".

This concludes the presentation of the theory of model specification and evaluation in practical situations. The next section is an example which illustrates this theory.
4.5 Example IV --- A Sophisticated Model.

In this section a model structure will be defined which is considerably more sophisticated than that of Example III (section 4.3). First a set of clauses, in \( L_c \), will be given for a group theory problem as it might be presented as an input set of clauses to a theorem prover. Then a \( \Sigma_f \) is given as a set of clauses in a language \( L_m \), where \( L_m \) is a language suitable for axiomatizing some of the properties of the real numbers considered as an ordered additive group. Then \( L_c, L_m, \Sigma_f \) (and an obvious \( \Gamma \), which will be given later) comprise the model structure we designate as \( M_f \). Some evaluations of sets of literals are illustrated by the use of \( \psi_{\Sigma_f} \) (of Theorem 4.2-8) based on \( M_f \). It is then argued that such an evaluation mechanism is computationally impractical and an alternative procedure is described. This alternative is a procedure, \( \Theta_{SL} \), based on simultaneous linear equations which will be used in the manner presented already in Theorem 4.4-2 and section 4.4.7. It is indicated by examples what the notion of a strengthening sequence can look like for a procedure such as \( \Theta_{SL} \). Finally the soundness of the semantic functions based on \( \Theta_{SL} \) is proved.
We consider the following set of clauses in $L_C$, which we call clause set $G$.

\[ \begin{align*}
G_1. & \quad P(x,1,x); \\
G_2. & \quad P(1,x,x); \\
G_3. & \quad P(g(x),x,1); \\
G_4. & \quad P(x,g(x),1); \\
G_5. & \quad P(x,y,f(x,y)); \\
G_6. & \quad \neg P(x,y,z), \neg P(z,u,v), \neg P(y,u,w), P(x,w,v); \\
G_7. & \quad \neg P(x,y,z), \neg P(x,w,v), \neg P(y,u,w), P(z,u,v); \\
G_8. & \quad P(x,x,1); \\
G_9. & \quad P(a,b,c); \\
G_{10}. & \quad \neg P(b,c,a); \\
G_{11}. & \quad \neg P(x,y,z), \neg LT(x,u), \neg P(u,y,v), LT(z,v); \\
G_{12}. & \quad \neg P(x,y,z), \neg LT(y,u), \neg P(x,u,v), LT(z,v); \\
G_{13}. & \quad EQ(x,x); \\
G_{14}. & \quad \neg EQ(x,y), EQ(y,x); \\
G_{15}. & \quad \neg EQ(x,y), \neg EQ(y,z), EQ(x,z); \\
G_{16}. & \quad \neg LT(x,x); \\
G_{17}. & \quad \neg LT(x,y), \neg LT(y,z), LT(x,z); \\
G_{18}. & \quad LT(x,y), LT(y,x), EQ(x,y); \\
G_{19}. & \quad \neg LT(x,y), \neg LT(y,x); \\
G_{20}. & \quad \neg EQ(x,u), \neg P(x,y,z), P(u,y,z); \\
& \quad \vdots \\
& \quad \vdots \\
& \quad \vdots \\
\end{align*} \]

where the axioms not explicitly written are the substitution axioms for equality, of which $G_{20}$ is one example. We see
that

\[\text{desc}(L_c) = \langle P(-,-,-), \text{LT}(-,-), \text{EQ}(-,-),
\langle 1, a, b, c, f (-,-), g (-) \rangle \rangle.\]

This clause set could be an axiomatization for the theorem:

"In an ordered group, if the square of every element is the identity element, then the group is commutative",

in which case the intended interpretation for the symbols in \(L_c\) is:

- \(P(x,y,z) \iff x \ast y = z\), where \(\ast\) is the group binary operation.
- \(\text{LT}(x,y) \iff x\) is less than \(y\)
- \(\text{EQ}(x,y) \iff x\) is equal to \(y\)
- \(g(x) \iff \) the element which is the inverse of \(x\) for \(\ast\).
- \(f(x,y) \iff x \ast y\)
- \(l \iff \) the group identity element
- \(a, b, c \iff \) any elements of the group such that \(a \ast b = c\) and \(b \ast a = c\).

We will set up a model structure, \(M = \langle L_c, L_m, T, \leq \rangle\) which intuitively captures most, but not necessarily all, of the usual properties of the real numbers considered as an ordered additive group.
We take
\[
\text{desc}(L_m) = \langle \{P', (-, -, -), <(-, -), =(-, -)\}, \\
\{0, a', b', c', +(-, -), -( -)\} \rangle,
\]
and the translation function, \( T \), will be induced by \( \mathcal{Y} : \)
\[
\mathcal{Y} - \{ \langle P, P' \rangle, \langle L T, < \rangle, \langle E Q, = \rangle, \\
<1, 0>, <a, a'>, <b, b'>, <c, c'>, <f, +>, <g, -> \}
\]
and "\(<, -, 0, +, -\)" are to be thought of as having their usual meaning on the real numbers. The meaning of \( P' \) is that for any terms \( t_1, t_2, t_3 \) of \( L_m \),
\[
P'(t_1, t_2, t_3) \iff t_1 + t_2 = t_3.
\]
\( P' \) is included in \( L_m \) only for the purpose of facilitating the translation from \( L_c \) to \( L_m \) (i.e. to make \( L_m \) amicable to \( L_c \)).

In distinction to Example III, here \( \mathcal{M}_L^{L_m} \) will not be a singleton set. This may be thought of as occurring because we do not assign specific values to \( a', b' \) and \( c' \).

For the scheme defining set we take initially
\[
\Xi_f = \{ g_1, g_2, \ldots, g_n \}
\]
where each \( g_i \) is a clause in \( L_m \). We require that \( \Xi_f \) be a consistent set of clauses, and capture most of the usual meaning of the symbols of \( L_m \) for the real numbers, but we are not particularly concerned with the independence of all of the clauses. Neither are we concerned that interpretations other than the real numbers satisfy \( \Xi_f \) (e.g. the rationals). It is intended that \( \Xi_f \) be a finite set of clauses, containing on the order of 30 clauses.
We indicate the structure of $\Sigma_p$ as follows:

$\Sigma_1$: \quad $P'(x,y,z) = (+(x,y),z)$;

$\Sigma_2$: \quad $P'(x,y,z), \neg = (+(x,y),z)$;

(And no other clause in $\Sigma_p$ uses the symbol $P'$)

$\Sigma_3$: \quad $=(x,x)$;

$\Sigma_4$: \quad $\neg=(x,y), \neg=(y,x)$;

$\Sigma_5$: \quad $\neg=(x,y), \neg=(y,z), =(x,z)$;

$\Sigma_6$: \quad $\neg=(+(x,y),z), \neg=(x,u), =(+(u,y),z)$;

(substitution axioms for $\neg$)

$\Sigma_7$: \quad $=+(x,0),x)$;

$\Sigma_{11}$: \quad $=+(x,-(x)),0)$;

$\Sigma_{12}$: \quad $=+(x,y), +y,x)))$;

(properties of $+, -, 0$)

$\Sigma_i$: \quad $<(x,y),<(y,x), =(x,y)$;

$\Sigma_{i+1}$: \quad $<(0,x),<(x,0)$;

(properties of $<$)

$\Sigma_d$: \quad $<=$(x,x)$;
Suppose \( Q \) is a clause (set of literals) in \( L_m \). We write \( \delta^{P'}(Q) \) to mean the result of resolving on all literals in \( Q \) which use the relation symbol \( P' \) with clause \( \delta_1 \) or \( \delta_2 \) (whichever is appropriate). We informally use the same notation even when \( Q \) is not a clause, e.g., if

\[
Q = \exists x: P'(x, x, a) \land \neg P'(b, x, a),
\]

then

\[
\delta^{P'}(Q) = \exists x: =+(x, x), a) \land \neg=(+(b, x), a).
\]

Then the intention is that \( \Xi_f \) will contain all of the following clauses:

1. \( \delta_1 \) and \( \delta_2 \);
2. \( \delta^{P'}(T(G_c)) \) for each clause \( G_c \) in clause set \( C \), for \( i = 1, 2, \ldots, 7, 11, 12, \ldots, 20 \).
3. clauses such as \( \delta_{41} \) and \( \delta_{47} \), listed above which are true for the real numbers, and
   i. contain no occurrences of \( P' \);
   ii. contain no occurrences of \( a', b' \), or \( c' \), nor introduce any symbols not in \( L_m \) already.

Notice that \( \Xi_f - \{ \delta_1, \delta_2 \} \) is a satisfiable theory of \( L_m \) since by its construction all of its clauses are true for the real number system. If we agree to interpret the predicate symbol \( P' \) such that

\[
P'(x, y, z) \text{ iff } =+(x, y), z)
\]

then \( \Xi_f \) is true in the real number system, and thus is a satisfiable theory of \( L_m \).
When evaluating a set of literals in $L_c$ by translating the negation into $L_m$, we obtain a conjunction of the negation of the literals with existentially quantified variables. If we convert this conjunction into clause form we get a set of unit clauses, $\{u_1, u_2, \ldots, u_n\}$. It should be clear, because of the structure of $\Sigma_f$, that $\{u_1, \ldots, u_n\} \cup \Sigma_f$ is satisfiable iff $$\langle \varphi_{\overline{\varphi}}(u_1), \ldots, \varphi_{\overline{\varphi}}(u_n) \rangle \cup (\Sigma_f - \{s_1, s_2\})$$ is satisfiable.

Assuming that $\Sigma_f$ has been made specific, we have now determined a particular model structure

$$M_i = \langle L_c, L_m, T, \Sigma_f \rangle$$

and we can consider various semantic functions for $L_c$ based on $M_i$.

We will illustrate some evaluations of sets of literals according to the semantic function $\psi_{\Sigma_f}$ of Theorem 4.2-8.

For $G_1$ we have

$$T(\neg G_1) \cup \Sigma_f = \{ \exists x: \neg \varphi'(x, 0, x) \} \cup \Sigma_f$$

and this can be shown to be unsatisfiable by converting to clause form:

$$\{ \neg \varphi'(d, 0, d); \} \cup \Sigma_f$$

(where $d$ is a new Skolem constant) and finding a refutation. We resolve $\neg \varphi'(d, 0, d)$ with $s_2$ to obtain $\neg =_{+(d, 0), d}$; and resolve this with $s_4$ to obtain the null clause. Thus we have $\psi_{\Sigma_f}(G_1) = true$.  

If we let $G_i'$ be the corresponding NLR input clause to $G_i$, i.e.

$$G_i' = \langle S(G_i'), \{\}\rangle$$

$$G_i = \lambda(S(G_i'))$$

then we can compute a t.v.d. value for $G_i'$ in a simple manner by noting the following:

From Theorem 4.2-9 and the definitions of $\alpha$- and $\beta$-statements of clauses, it can be shown that if a clause has an empty FSL set, and $\psi_f^c$ on its standard literals is true, then the t.v.d. value of the clause is $T$. Also, if the FSL is empty, $\psi_f^c$ on the standard literals is false iff the $\alpha$-statement of the clause is false under $\psi_f^c$.

For the clause $C_6$ we have

$$\mathcal{I}(\sim \text{C}_6) \cup \mathcal{E}_f = \{ \exists x, y, z, u, v, w : P'(x, y, z)$$

$$\wedge P'(u, v) \wedge P'(y, u, w) \wedge \sim P'(x, w, v) \} \cup \mathcal{E}_f$$

and this is satisfiable iff

$$Q = \{ \equiv (+e, f, g); \equiv (+g, h, i);$$

$$\equiv (+f, h, j); \equiv (+e, j, i); \} \cup \mathcal{E}_f$$

is satisfiable, where $e, f, g, h, i, j$ are new Skolem constants.

Now, by our construction of $\mathcal{E}_f$, the clause $\psi_f^c \cup \mathcal{I}(\text{C}_6)$ is in $\mathcal{E}_f$, so that $Q$ will be unsatisfiable, and thus $\psi_f^c(C_6) = \text{true}$, and t.v.d. $(C_6', \psi_f^c) = T$. By the same reasoning

$$\psi_f^c(G_i) = \text{true} \text{ and } \text{t.v.d.} (G_i', \psi_f^c) = T$$

for $i = 1, 2, \ldots, 7, 11, 12, \ldots, 20 \ldots$. 
For clause C8 we have
\[ \forall \neg C8 \cup \Sigma_g = \{ \exists x: \neg p'(x, x, 0) \} \cup \Sigma_g \]
which when put into clause form (and \( g_x \) is used to replace the "\(-\)" literal with a "\(-\)" literal) gives
\[ \{ \neg+(k, k), 0 \} \cup \Sigma_g \]
with \( k \), a new Skolem constant. This set of clauses is satisfiable, so we have \( \psi_g(C8) = \text{false} \). The corresponding HLR input clause, \( C8' \), has no FSL literals, so that
\[ \psi_g(C8') = \psi_g(C8) = \text{false} \].

To compute the t.v.d. value of \( C8' \) we require the \( \varnothing \)-statement truth value under \( \psi_g \), which we obtain as follows.
\[ \psi_g(\varnothing(C8')) = \psi_g(\forall x: \neg p'(x, x, 1)) = \psi_g(\neg p(x, x, 1)) \]
Thus we form
\[ \forall (\neg p'(x, x, 1)) \cup \Sigma_g = \{ \exists x: p'(x, x, 0) \} \cup \Sigma_g \]
which in clause form is
\[ \{ +(m, m), 0 \} \cup \Sigma_g \]
for \( m \) a new Skolem constant. This set is satisfiable, so that
\[ \psi_g(\varnothing(C8')) = \text{false} \]
and thus \( \text{t.v.d.}(C8', \psi_g) = T/F \).

In a similar fashion we obtain
\[ \text{t.v.d.}(C9', \psi_g) = T/F \]
and
\[ \text{t.v.d.}(C10', \psi_g) = T/F \]
Note that even though G9 and G10 are opposite sign ground literals, and even though (because of ξ₂,₂ along with other clauses in Σ₂) the relation symbol P is interpreted as symmetric on its first two terms, they both are false sets of literals according to ψ². This is because in some of the interpretations in m², G9 is true (and G10 is false), and in the rest G9 is false (and G10 is true).

As can be seen from the above examples of ψ² evaluations on input clauses, there is no difficulty in doing these by hand. Because of the close relationship between G and Σ₂, it is also possible to specify a general purpose theorem proving procedure, R, such that a semantic function ψ², as in Theorem 4.4.1, would be of quite modest cost to evaluate for clauses in G. However, for clauses derived from G (during a resolution search process) there is no reason to expect that simple general purpose procedures will be pragmatically effective. The reason is that Σ₂ is too prolific, consisting of roughly 30 clauses, averaging just over 2 literals per clause, and axiomatizing equality explicitly. If a reasonable size effort cutoff is incorporated in R, then many unsatisfiable clause sets would not be detected as unsatisfiable, and ψ² will be false for many more sets of literals in Le than ψ². This will increase the number of clauses at each level of the search.
We will now describe a procedure, $\mathcal{P}_{SLE}$, which is not a general-purpose theorem proving procedure, and is a reasonably efficient procedure for detecting unsatisfiability of the sets of literals in which we are interested in $L_{\mathcal{M}}$. This procedure has been implemented and has performed well in the following two senses:

1. The procedure is incomplete, but in practice fails to detect unsatisfiability only in a very small percentage of the cases which are unsatisfiable;

2. Although there is no specific cutoff for the amount of work allowed, the procedure is an algorithm in that it terminates either by detecting unsatisfiability or by attaining a configuration from which no change is possible. The amount of effort involved in reaching a terminating condition has been modest in pragmatic situations.

$\mathcal{P}_{SLE}$ is a procedure for detecting unsatisfiability of a set of Simultaneous Linear Equations over the real numbers, and consists of two main components: an algorithm, $A$, for testing a set of equations for consistency (i.e., satisfiability over the real numbers), and a sentence of $L_{\mathcal{M}}$, $C$, called the context set. The context set consists of a sentence of $L_{\mathcal{M}}$ which is of the form

$$\exists x_1 \exists x_2 \ldots \exists x_j : Z_1 \land Z_2 \land \ldots \land Z_m$$

where each $Z_k$ is a single literal using the relation symbol "$\leq$" or "$\geq$" (possibly negated), and $x_1, \ldots, x_j$ are all of the variable symbols in $Z_k$'s. We will use $\mathcal{P}_{SLE}$ in the
manner indicated by Theorem 4.4-2 and section 4.4.7, and leave until later the question of the soundness of the resulting semantic function. In what follows the description of $\mathcal{E}_{\mathcal{L}}$ is essentially the same as an implemented version of $\mathcal{E}_{\mathcal{L}}$ which was used experimentally.

Notice that $L_c$ and $T$ are such that for $K$ any set of literals in $L_c$, $\theta_{\mathcal{E}}(T(\neg K))$ is a conjunction containing only existentially quantified variables, and only having the relation symbols "=" and "<". The algorithm, $A$, requires first that the sentence in $L_{\mathcal{M}}$ be re-written so that the only predicate symbol is "=" and $A$ also requires that this does not occur negated. To do this we assume the existence of 3 infinite sets of symbols, called class constants, which are denoted by $\Delta_c^\circ$, $\Delta_c^\circ$ and $\Delta_c^\circ$, for $i$ any positive integer. The intended meaning of $\Delta_c^\circ$, for some specific $j$, is that it represents some real number greater than zero. Similarly for $\Delta_c^\circ$ and $\Delta_c^\circ$. In the following $L_c$, $L_{\mathcal{M}}$ and $T$ are as determined previously for $M_I$. We assume, whenever required for expressions in $L_{\mathcal{M}}$, that literals using $P'$ are replaced by the appropriate equality literals, even when we do not write the "$\mathcal{J}_{\mathcal{M}}/P'$" notation explicitly.
PROCEDURE IV -- 1

Procedure for Forming Equation Set $K_{\phi'}$.

Let $K$ be a sentence (of $L_m$ as defined for $K_{\phi}$) of the form

$$\exists x_1 \exists x_2 \ldots \exists x_i : k_1 \land k_2 \land \ldots \land k_n$$

where each $k_i$ is a literal using the relation symbol "=" or "<" (possibly negated), and $x_1, x_2, \ldots, x_i$ are all of the variable symbols in the $k_i$'s. We denote by $K_{\phi'}$ the set of equations formed by the following steps applied to $K$:

1. Remove the existential quantifiers, leaving the conjunct with free variables.
2. For $t_1, t_2$ any terms of $L_m$, and for $i = 1, 2, \ldots, n$:
   - If $k_i$ is $= (t_1, t_2)$, then we replace it with $t_1 = (t_2) = 0$.
   - If $k_i$ is $<(t_1, t_2)$, then we replace it with $t_1 - (t_2) + \Delta_{t} = 0$, where $r$ is a positive integer which has not previously been used for a class constant subscript in any equation formed for $\Theta_{SLE}$.
   - If $k_i$ is $>(t_1, t_2)$, then replace it with $t_1 - (t_2) + \Delta_{s} = 0$, where again $s$ is a new positive integer.
   - If $k_i$ is $\neg<(t_1, t_2)$, then replace it with $t_1 - (t_2) - \Delta_{t} = 0$, where $t$ is a new positive integer.
3. Eliminate the conjunction signs and consider the equations as a set, $K_{\phi'}$. 


The result, $K_{\mathcal{F}}$, is clearly not in the language $I_{\mathcal{F}}$.

Notice that the usual meaning of "is an inconsistent set of equations" does coincide with the existential interpretation of the variable symbols. The algorithm $A$ must treat variables in the intended way, as well as class constants. The remaining symbols in $K_{\mathcal{F}}$ are to be interpreted in the usual way over the domain of real numbers. We assume all the usual amenities of algebraic manipulations, so that, for example, if the equation

$$+(+(x,a'),z) - (-(z)) + \Delta^{\wedge}_{3} = 0$$

is an equation in $K_{\mathcal{F}}$, it can be re-written as

$$x + 2z + a' + \Delta^{\wedge}_{3} = 0$$

Now if $A$ applied to $(\mathcal{T}(\text{~K})_{\mathcal{F}} \cup C_{\mathcal{F}}$ is inconsistent, i.e. $A((\mathcal{T}(\text{~K})_{\mathcal{F}} \cup C_{\mathcal{F}}) = inconsistent$, then we say $\theta_{\text{true}}(\mathcal{T}(\text{~K})) = inconsistent$, and $\psi_{\text{true}}(K) = true$ as in Theorem 4.4-2. Otherwise we take $\theta_{\text{true}}(\mathcal{T}(\text{~K})) = unknown$ and $\psi_{\text{true}}(K) = false$.

We now outline the algorithm $A$. An ordering, $\prec_{\mathcal{F}}$, which is a simple ordering of all the variable and constant symbols in $(\mathcal{T}(\text{~K})_{\mathcal{F}} \cup C_{\mathcal{F}}$ is chosen such that all variable symbols precede all constant symbols, and all ordinary constants precede all class constants. It is optional as to whether a single ordering is chosen initially to cover all sets of equations, or the ordering is changed for different sets of equations. $A$ is applied to a set of equations, $G$, as follows.
Algorithm Scheme For A

(A applied to equation set Q, given ordering \(<_A\))

**Step 1.** For each \(q\) in \(Q\), the terms are rearranged so that under \(<_A\) they are in increasing order from left to right (a term such as "nx", for \(n\) an integer, is equivalent to "x" in the ordering). go to step 2.

**Step 2.** The resulting set of equations is put into Hermite normal form (also known as (reduced) row echelon form); since \(Q\) contains only integer coefficients we can do the equation manipulations entirely in the integers. go to step 3.

**Step 3.** The resulting set is checked for obviously inconsistent single equations such as:

for \(i, j\) any positive integers,

\[
\begin{align*}
&i. \quad n \Delta_j^o = 0 \\
&\text{or} \\
&\quad \text{ii.} \quad n \Delta_j^x = 0 \\
&\quad (\text{for } n \neq 0) \\
&\text{or} \\
&\quad \text{iii.} \quad n \Delta_j^o + m \Delta_j^x = 0 \\
&\quad (\text{for } n \cdot m > 0)
\end{align*}
\]

and so on.

If an inconsistent equation is found then \(A\) terminates with the value "inconsistent". If not, go to step 4.

**Step 4.** Each equation is individually processed by attempting to match it with a re-write rule, such as

\[
n \Delta_i^o + m \Delta_j^o = 0
\]

\[
\Rightarrow
\]

\[
\Delta_i^o = 0
\]

\[
\Delta_j^o = 0
\]

for \(n, m, i, j\) any positive integers, which allows the equation to be replaced by simpler equations. If no equation can be matched, then \(A\) terminates and returns the value "unknown". If there are matches then all possible re-write rules are applied, and go to step 2.

end of algorithm A.
By interpreting all of the symbols in the usual way (and for class constants in the obvious way as existentially quantified variables with a restricted range of quantification), we have a well defined notion of unsatisfiability of a set of equations, $K_x$, over the reals. Then we say that $A$ is complete iff for all (finite) sets of equations, $K_x$, $A(K_x)$ = inconsistent whenever $K_x$ is unsatisfiable over the real numbers. The above procedure for $A$ is not complete. The main reason for the incompleteness of $A$ is that the choice of an ordering for the terms and the process of converting to Hermite normal form does not always give a single equation which is unsatisfiable, even when the entire set is unsatisfiable (over the reals) and no re-write rules are applicable. The implementation of $R_{cl}$ was incomplete for precisely this reason. A second source of incompleteness is when, in step 3 of the algorithm scheme for $A$, not all individually unsatisfiable equations are recognized. Similarly in step 4 the set of re-write rules may not be "complete".

In order to show that $A$ will always terminate, the specific re-write rules of step 4 must be known. In the version of $A$ actually implemented termination was easy to prove.
We illustrate the evaluation process for the clause G11, using a context set which is the empty conjunction.

\[ \text{E}_{\text{eval}}(\Gamma(\neg G11)) = \{ \exists x, y, z, u, v: -(+(x, y), z) \land \less (x, v) \land -(+(u, y), v) \land \neg \less (z, v) \} \]

becomes the set of equations, G11_\text{eval}:

\[
\begin{align*}
    x + y - z &= 0 \\
    x - u + \Delta_i^0 &= 0 \\
    u + y - v &= 0 \\
    z - v - \Delta_2^0 &= 0
\end{align*}
\]

and since \( C = \{ \} \), this is the set that \( A \) operates on. We take

\[ x \less x \less y \less z \less u \less v \less \Delta_i \less \Delta_2. \]

The Hermite normal form is then

\[
\begin{align*}
    x - u - \Delta_2^0 &= 0 \\
    y + u - v &= 0 \\
    z - v - \Delta_2^0 &= 0 \\
    \Delta_i^0 + \Delta_2^0 &= 0
\end{align*}
\]

and we assume that step 3 of \( A \) can recognize this last equation as inconsistent. Thus \( \text{eval}(G11) = \text{true} \), where we use the superscript of \( G_{\text{MLE}} \) to indicate the context set. In terms of the notation already introduced we explicitly define
\[ \gamma_{\mathcal{D}_{\Sigma}}(K) = \begin{cases} 
\text{true} & \text{if } \mathcal{A}(\{\overline{1}, \overline{K}\})_{\Sigma} \cup \alpha_{\Sigma} = \text{inconsistent} \\
\text{false} & \text{otherwise} \end{cases} \]

To evaluate \( \gamma_{\mathcal{D}_{\Sigma}}(G9) \) we have

\[ \delta_{\mathcal{D}_{\Sigma}}(\{\overline{1}, \overline{G9}\}) = \delta_{\mathcal{D}_{\Sigma}}(\{\overline{1}, \overline{\{P(a,b,c)\}}\}) = \{ \overline{=}+(a',b'),c' \} , \]

and this gives the equation set

\[ a' + b' - c' + \Delta_3^{\Sigma} = 0 . \]

A applied to this will terminate with the value "unknown", so we have \( \gamma_{\mathcal{D}_{\Sigma}}(G9) = \text{false} \). In a similar manner we find that \( \gamma_{\mathcal{D}_{\Sigma}}(G10) = \text{false also} \).

Now suppose we change the context set to

\[ C_i = +(a',b'),c' . \]

Then the set of equations that \( A \) acts on for the evaluation of \( G9 \) is

\[ a' + b' - c' + \Delta_3^{\Sigma} = 0 \quad \text{(from \( G9 \))} \]
\[ a' + b' - c' = 0 \quad \text{(from \( C_i \))} \]

and the Hermite normal form is

\[ a' + b' - c' = 0 \]
\[ \Delta_3^{\Sigma} = 0 \]

and, assuming step 3 of \( A \) can detect the inconsistency of this set, we now have \( \gamma_{\mathcal{D}_{\Sigma}}(G9) = \text{true} \). \( \gamma_{\mathcal{D}_{\Sigma}}(G10) \) is false, as before.

Thus we see that the procedure \( \mathcal{D}_{\Sigma} \) can be modified, by changing the context set, so as to make \( \gamma_{\mathcal{D}_{\Sigma}} \) evaluate some sets of literals as true which were previously false. The
requirement for \( \gamma e_{\text{SLE}} \) to be a sound function is that \( C \) must
be a satisfiable sentence in the real numbers (that this is
so should become clear in the remainder of this section).

To evaluate clause G8 according to \( \gamma e_{\text{SLE}} \) we take
\[
\mathcal{N}_{\mathcal{F},/}(\overline{\mathcal{A}}(\neg G8)) = \mathcal{N}_{\mathcal{F},/}(\overline{\mathcal{A}}(\neg \{P(x,x,1)\}))
\]
\[
= \{ \neg = (+ (x,x), 0) \}
\]
and the equation set is
\[
\begin{align*}
2x + \Delta^0_x = 0 \\
 a' + b' - c' = 0
\end{align*}
\]
which is consistent, so \( \gamma e_{\text{SLE}}(G8) = \text{false} \). In this case
there is no meaningful way to modify the context set so as
to make the evaluation of G8 be true. The reason is that
the context set is global to all the evaluations while the
symbol "x" in the above set is local to this particular
evaluation. Therefore the context set would have to include
something equivalent to "\( \forall y: y = 0 \)" in order to make the G8
evaluation be true. But this would make the context set
unsatisfiable over the reals. This is why the context sets
are restricted to be pure existential in prenex form.

For example, instead of changing the context set from
\( C = \text{"empty conjunction"} \) to
\( C_e = (+ (a', b'), c') \),
as we did before, thus making G9 true and G10 false, we
instead could have taken the context set to be
\( \neg = (+ (a', b'), c') \),
which would make G9 false and G10 true.
Now we consider the HLR input clause, $G8'$, corresponding to $G8$. Since $\forall \sigma(G8') = \{\}$, we have

$$\gamma^{P_{\mathcal{L}}}C_{i}(\forall(G8')) = \text{false,}$$

(because $\forall(G8') = \forall x: \Box \rightarrow P(x,x,1)$, and

$$\gamma^{P_{\mathcal{L}}}C_{i}(\{P(x,x,1)\}) = \text{false}.$$  

Also we have

$$\gamma^{P_{\mathcal{L}}}C_{i}(\Box(G8')) = \gamma^{P_{\mathcal{L}}}C_{i}(\forall x: [A] \rightarrow \neg P(x,x,1))$$

$$= \gamma^{P_{\mathcal{L}}}C_{i}(\{\neg P(x,x,1)\}) = \text{false}$$

(because the resulting equation set

$$2x = 0 \quad \text{(from $\neg P(x,x,1)$)}$$

$$a' + b' - c' = 0 \quad \text{(from $C_1$)}$$

is consistent).

Thus we have $t.v.d.(G8', \gamma^{P_{\mathcal{L}}}C_{i}) = T/F$.

The remainder of this example consists of a demonstration of the soundness of $\gamma^{P_{\mathcal{L}}}C_{i}$, and of the indexed soundness of $\gamma^{P_{\mathcal{L}}}C_{i}$ (when $\mathcal{G}$ is a particular type of sequence of context sets). In the demonstration, $L_\epsilon$, $L_m$, and $T$ are as in $M_1$. We will require also a new language of the model, $L_m^*$, which is an expansion of $L_m$, formed by adding the 3 infinite sets of class constant symbols to $L_m$ along with a two place function, "*", and a constant "1". This gives $L_m^*$ as a language suitable for axiomatizing the real number system with order. The essence of the proof of soundness is to show the existence of a satisfiable theory, $\Sigma^{C,m}$, of $L_m$, such that $\gamma^{P_{\mathcal{L}}}C_{i}$ is false permissive wrt. $\gamma^{P_{\mathcal{L}}}C_{i}$. Since $< L_\epsilon, L_m, T, \Sigma^{C,m} >$ is a model
structure, \( \psi^{e_m} \) will be a sound semantic function for \( L_e \),
and thus so will \( \psi^{e_{iE}} \) be a sound semantic function for \( L_e \).
A variation of this argument then allows the proof of the
soundness of the indexed semantic function \( \psi^{e_{iE}} \).

We start by writing our \( \psi^{e_{iE}} \) definition in a slightly
different form than before:

\[
\psi^{e_{iE}}(K) = \begin{cases} 
\text{false if } & \theta_{iE}^e(\neg K) = \text{unknown} \\
\text{true if } & \theta_{iE}^e(\neg K) = \text{inconsistent}
\end{cases}
\]

where we define \( \theta_{iE}^e(\neg K) \) to have the value
\( A((\neg K) \cup C_{\phi^e}) \), and \( C_{\phi^e} \) is explained immediately below.
Now \( \psi^{e_{iE}}(K) \) is analogous to the form of \( \psi^{e} \) in Theorem
4.4-2, i.e. \( C \) has been made a part of the procedure
designation. \( C \) is a pure existential sentence in prenex
normal form, in \( L_m \), using only relation symbols "-" or "<",
or their negations, and with a matrix which is a conjunction
of literals. Thus Procedure IV - I can be applied to \( C \) to
give \( C_{\phi^e} \) as a set of equations, possibly including class
constants. While we consider these as a set of equations,
they are of course (each set) equivalent to a sentence in
\( L_m^{*} \). The sentence in \( L_m^{*} \) that corresponds to \( C_{\phi^e} \) (and
corresponds to \( C \), a context set in \( L_m \)) has a matrix that is
a conjunction of equality literals (one for each equation in
\( C_{\phi^e} \)), and is in prenex normal form with a prefix consisting
of existential quantifiers for all of the variables.
We designate by REAL the real number system. Let $m^*$ be an assignment of real numbers to (all of) the constants in $L_{m^*}$, and $m$ be the restriction of $m^*$ to the constants of $L_m$. (N.B. The only constants in $L_m$ are $0$, $a^*$, $b^*$, and $c^*$, while $L_{m^*}$ has these plus all of the class constants and "1"). We say $m^*$ is a standard mapping for $L_{m^*}$ iff

1. $m^*$ assigns the real number $0$ to the symbol "0";
2. $m^*$ assigns the real number $1$ to the symbol "1";
3. for each positive integer $i$, $\Delta^*_i$ is assigned a non-zero real number, $\Delta^*_0$ is assigned a positive non-zero real number, and $\Delta^*_i$ is assigned a non-negative real number.

We say the restriction $m$, of $m^*$ to $L_m$, is a standard mapping for $L_m$ if condition 1 above holds.

Let $\text{REAL}_{m^*}$ be a relational structure on the real numbers for the language $L_{m^*}$ in which the constant symbols of $L_{m^*}$ are assigned real numbers according to the standard mapping $m^*$, and

1. $<, =, +, -$ are all assigned relations or functions on the real numbers which are the obvious ones indicated by the symbols;
2. $P'$ is assigned the relation given by
   \[ P'(x, y, z) \text{ iff } x + y = z. \]
Then we say that $\text{REAL}_m^*$ is a standard relational structure for $L_m^*$. We let $\text{REAL}_m$ be the restriction of $\text{REAL}_m^*$ to the language $L_m$, and say it is a standard relational structure for $L_m$ if $m$ is a standard mapping for $L_m$.

We say that a sentence, $s^*$, in $L_m^*$ is satisfiable over the reals iff

$$\exists m^*: \left[ (m^* \text{ is a standard mapping for } L_m^*) \right. \left. \land \right. \left. (\text{REAL}_{m^*} \text{ is a standard relational structure for } L_m^*) \right. \left. \land \right. \left. (s^* \text{ is true in REAL}_{m^*}) \right]$$

(where logical symbols in $s^*$ are interpreted in the usual manner). Similarly for $s$ a sentence in $L_m$, $s$ is satisfiable over the reals iff

$$\exists m: \left[ (m \text{ is a standard mapping for } L_m) \right. \left. \land \right. \left. (\text{REAL}_m \text{ is a standard relational structure for } L_m) \right. \left. \land \right. \left. (s \text{ is true in REAL}_m) \right]$$

A theory in $L_m^*$ is satisfiable over the reals iff there exists a standard relational structure, $\text{REAL}_{m^*}$, such that every sentence in the theory is true in $\text{REAL}_{m^*}$. A similar definition holds for a theory in $L_m$.

The property "satisfiable over the reals" has been given a definition which is intuitively the obvious one for both $L_m$ and $L_m^*$. Let $C$ be a context set in $L_m$ such that $C$ is satisfiable over the reals, and let $C_{eq}$ be the result of applying Procedure IV-1 to $C$. Then obviously $C_{eq}$ is
satisfiable over the reals (under the convention that $C_{\phi}$ is a purely existential sentence in prenex normal form with a matrix consisting of a conjunction of equality literals).

Let $m$ be a standard mapping for $L_m$ such that $C$ is true in the standard relational structure $\text{REAL}_m$. Define

$$\Sigma^{c_m} = \{ s \mid (s \text{ a sentence of } L_m) \land (s \text{ is true in REAL}_m) \}$$

$\Sigma^{c_m}$ is satisfiable and is a complete theory of $L_m$. Also note that $C \in \Sigma^{c_m}$. Since $L_c$, $L_m$, $\text{IT}^{\Sigma^{c_m}}$ is a model structure, we know $\gamma^{L_c^{c_m}}$ is a sound semantic function for $L_c$.

Suppose $K$ is a set of literals in $L_c$ such that $\gamma^{L_c^{c_m}}(K) =$ false. Then we know (from Theorem 4.2-8) that

$$(\text{satisfiable } (\text{T}^{\neg(K)}) \cup \Sigma^{c_m})$$

is true. But $\Sigma^{c_m}$ is a complete theory of $L_m$, therefore $\text{T}^{\neg(K)} \in \Sigma^{c_m}$. Therefore $\text{T}^{\neg(K)}$ is true in REAL$_m$. Thus $\text{T}^{\neg(K)} \cup C$ is true in REAL$_m$ (there are no shared variables between $\text{T}^{\neg(K)}$ and $C$, since they are each existentially closed). Therefore $\text{T}^{\neg(K)} \cup C$ is satisfiable over the reals, and so therefore is $(\text{T}^{\neg(K)})_{\exists} \cup C_{\exists}$. Further, there will exist an extension of $m$, $m^*$, such that $m^*$ is a standard mapping of $L_{m^*}$, and $(\text{T}^{\neg(K)})_{\exists} \cup C_{\exists}$ is true in the standard relational structure $\text{REAL}_{m^*}$. 

Now suppose that it can be shown that

\[ \left( \left( \{ \Gamma(\neg K) \}_\mathbb{R} \cup C_{\aleph_0} \right) \text{ true in } \text{REAL}_{\aleph_0^*} \right) \]

\[ \implies \left( A \left( \left( \{ \Gamma(\neg K) \}_\mathbb{R} \cup C_{\aleph_0} \right) \text{ unknown} \right) \right). \]

Then this gives \( \psi_\aleph^C(K) = \text{false} \), and we will have shown that \( \psi_\aleph^C \) is a sound semantic function for \( L_{\aleph^0} \).

We do this in outline as follows. Assume \( A \) has been shown to be terminating, and \( S_i = \left( \{ \Gamma(\neg K) \}_\mathbb{R} \cup C_{\aleph_0} \right) \). We represent the execution of procedure \( A \) starting with \( S_i \) as a finite sequence of equation sets, \( S_1, S_2, \ldots S_m \). There are two conditions to consider.

1. Suppose at some point in the execution of \( A \), equation set \( S_\ell \) is replaced by (changed to) equation set \( S_{\ell+1} \), then

\[ \forall \bar{m}^* : \left( \left( \bar{m}^* \text{ a standard mapping for } L_{\aleph^0} \right) \right) \]

\[ \left( S_\ell \text{ true in } \text{REAL}_{\aleph^0}^* \left( L_{\aleph^0} \right) \right) \]

\[ \implies \left( S_{\ell+1} \text{ true in } \text{REAL}_{\aleph^0}^* \left( L_{\aleph^0} \right) \right) \].

(where we require \( \text{REAL}_{\aleph^0}^* \) to be a standard relational structure). This means that if \( S_i \) is satisfiable over the reals, then so is \( S_\ell \), for \( i = 2, \ldots n \).

2. If \( A \) recognizes an equation, \( E \), in equation set \( S_\ell \), as inconsistent, then there exists no standard mapping for \( L_{\aleph^0}^* \), \( \bar{m}^* \), such that \( E \) is true in the standard relational structure \( \text{REAL}_{\aleph^0}^* \).
These two conditions have to be verified for the algorithm A. If they are true then the soundness of \( \chi_{\delta_{Le}}^{c} \) follows.

Since A has not been given in detail, we cannot explicitly verify the above two conditions. In the implemented version of \( \delta_{Le} \) the procedure A used rules of manipulation of equations and rules for recognizing the inconsistency (i.e., unsatisfiability over the reals using standard relational structures for \( L_{\mathbb{R}}^{k} \)) of equations which obviously met the two conditions above.

Assuming that A meets the conditions, then what we have in effect shown is the truth of the \( \Sigma^{c,m} \)-justification statement for \( \chi_{\delta_{Le}}^{c} \) in \( L_{\mathbb{R}} \), namely:

\[
\forall s: [ (s \text{ a sentence of } L_{\mathbb{R}}) \implies \left( ( \chi_{\delta_{Le}}^{c}(s) = \text{inconsistent}) \implies (\text{unsatisfiable } (s \cup \Sigma^{c,m})) \right) ]
\]

Note that we can write "\( \Sigma^{c,m} \)" in the above, instead of "\( \Sigma^{c,m} \cup C \)" because \( C \subseteq \Sigma^{c,m} \).

This completes the demonstration of the soundness of \( \chi_{\delta_{Le}}^{c} \) for a given fixed \( C \) which is satisfiable over the reals.
The reader should be able to see that the above demonstration in no way depends upon the intended meaning of the clauses in $L_c$, and also is independent of the nature of $T$, except insofar as $T$ is required to translate $L_c$ into $L_m$. Thus the demonstration of soundness really just demonstrates a property of $\phi_{c/\varepsilon}$ relative to the language $L_m$ as defined for $M_i$. As such we can assert the soundness of $\psi_{c/\varepsilon}$ for any language $L_c$ and using any translation function $T'$, such that $T'$ translates $L_c$ into $L_m$ (and assuming $C$ is satisfiable over the reals, of course).

We now look at the case where we wish to change the context set during the process of expanding the search space in $L_c$. We do this in outline only.

Let $C = C_1, C_2, C_3 \ldots$ be a strengthening sequence of context sets, such that the matrix of $C_i$ is a subset of the matrix of $C_j$, when $i < j$, and such that

\[\exists j : (\text{for all positive integers } i, C_i \text{ true in } \text{REAL}_{\varepsilon})\]

(whence $\text{REAL}_{\varepsilon}$ is a standard relational structure).
Then define, for $K$ a set of literals in $L_\varepsilon$ and for $i$ a positive integer,

$$
\phi^{\varepsilon_t}(K,i) = \begin{cases} 
\text{false} & \text{if } \gamma^{\varepsilon_t}(K) = \text{false} \\
\text{true} & \text{if } \gamma^{\varepsilon_t}(K) = \text{true}.
\end{cases}
$$

Then $\phi^{\varepsilon_t}(K,i)$ is a sound indexed semantic function for $L_\varepsilon$.

We show this as follows:

$$
\exists \varepsilon \forall i: (i \in \text{positive integers} \implies \varepsilon \text{ true in } \text{REAL}_{\varepsilon_{\infty}}).
$$

Then define $\Sigma^\varepsilon_{\infty}$ as the set of sentences of $L_{\varepsilon}$ that are true in $\text{REAL}_{\varepsilon_{\infty}}$. $\varepsilon$ is true in $\text{REAL}_{\varepsilon_{\infty}}$, and therefore is in $\Sigma^\varepsilon_{\infty}$ for every $\varepsilon$. By arguments roughly similar to what has been done previously, we show that

$$
\gamma^{\varepsilon_{\infty}}(K) = \text{false} \implies \forall i: \phi^{\varepsilon_t}(K) = \text{false}.
$$

Then $\gamma^{\varepsilon_{\infty}}(K) = \text{false} \implies \forall i: \phi^{\varepsilon_t}(K,i) = \text{false}$, and thus $\phi^{\varepsilon_t}$ is a sound indexed semantic function.

In practical cases the most likely way of changing from one context set to the next would be by adding more conjuncts (i.e. equations). However, the above argument covers all cases where

$$
\bigcup_{\varepsilon \in \mathcal{E}} C_\varepsilon
$$

is true in some standard relational structure for $L_{\varepsilon}$.
In summary we have shown in this example how a moderately complex set of clauses, \( G \), can be modeled by a sub-theory of the usual real number theory. The model structure for doing this was relatively simple, involving a finite scheme defining set, \( \Sigma \). The result is a sophisticated sound semantic function, \( \gamma^\Sigma \), which makes almost all of the clauses in \( G \) true and intuitively captures most of the relevant phenomenon represented syntactically by \( G \). Both the underlying model schemes (generated by \( \Sigma \)) and the individual Herbrand interpretations in the model schemes are non-trivial. It was also shown how a procedural approach to model evaluations, in the form of \( \gamma^{\text{proc}} \), could be accomplished. The soundness of the semantic function \( \gamma^{\text{proc}} \) was demonstrated (in outline) for a fixed context set \( C \), and the indexed soundness of \( \gamma^{\text{proc}} \) argued for \( G \), a strengthening sequence of context sets.

This concludes the treatment of Example IV.
4.6 General Discussion of Models.

It is the purpose of this section to provide a context and perspective on the technical presentation of the earlier sections of this chapter.

The reader is probably aware that the word "model" has not yet been defined. Instead "model scheme", "model structure" and "semantic functions" (of various types) have been defined.

Of these last three notions, semantic function is closest to what we mean by a model since the search process is affected by the model only through the corresponding semantic function. On the other hand when concentrating attention on the nature of a semantic function, such notions as model structure, model scheme and evaluation procedure become important. Thus we will use "model" as a non-specific term covering all of the previously discussed notions.

4.6.1 Rationale for Using Models.

Models are of potential utility in increasing theorem proving efficiency for at least two distinct reasons. The first is that they provide a search space organization different than that of syntactic strategies, and it is clear that (at least present day) syntactic strategies have serious defects. Reiter has argued (Reiter, 1972) that just
from the viewpoint of flexibility and comprehensiveness alone it is more advantageous to implement domain dependent information in a theorem prover by models rather than by syntactic methods.

The second reason has to do with the nature of the clause sets which are likely to arise in realistic situations. These clause sets are such that almost all of the clauses are simultaneously true in at least one interpretation. Consider a clause set with \( N \) clauses, and an interpretation in which a small fraction, \( f \), of the clauses are false. Then the number of potential parent pairs for forming clauses at level 1 is approximately \( f^* (N^{**2})/2 \) under TMS, compared to \( (N^{**2})/2 \) for most non-semantic based refinements. For many realistic clause sets arising in mathematical type problems, there are models such that \( f = c/N \), where \( c \) is a small integer (often \( c = 1 \)). In that case \( f^* (N^{**2})/2 = c^*N/2 \). Employing such a model with TMS then gives a level 1 size that is roughly linearly related to the input set size. At deeper levels the rate of growth will not be held in check so strongly through the effect of a small value for \( f \) since TMS tends to produce a relatively high fraction of false resolvents. However, notice that since every clause at levels deeper than level 1 has at least one level 1 ancestor, there will be a strong effect at all levels on the terms that appear in clauses due to the selectivity of TMS in forming level 1. In addition a strategy such as TMS (in distinction to set of support)
exerts a continuing effect at each level, directly, on the terms that are allowed to appear. This effect is even more pronounced when FSL's are incorporated into the strategy.

We now give a slightly different perspective on this second reason for using models. As more complex clause sets are considered, where complexity is defined by any one of (or combination of) common measures such as number of input clauses, number of relevant input clauses, number of relevant ground instances, term complexity, etc., a measure of complexity which tends to grow slowly is the number of input clauses that are false, $N_f$ (assuming good models are available for the domains being represented). For the special case of minimally unsatisfiable input sets, there will always exist some model such that $N_f = 1$. In realistic situations the input set is usually not even close to being minimally unsatisfiable, but yet often there are known models in which only a few clauses are false. Example IV (section 4.5) is an example of this. There the clause set $G$ (consisting of more than 20 clauses) has the subset

$$U = \{G_1, G_2, G_6, G_7, G_8, G_9, G_{10}\}$$

as an unsatisfiable subset. The model scheme defining set, $\Sigma_f$, used there in $\psi^{\Sigma_f}$, makes $G_8$, $G_9$ and $G_{10}$ false (and could be modified so as to make only $G_8$ and one of $G_9$ and $G_{10}$ false), and the rest of $U$ true. But notice that every clause in $G - U$ is also true according to $\psi^{\Sigma_f}$. Thus these clauses are not allowed to resolve with each other directly. Such a situation as this is not unlikely for realistic
clause sets and points out that even though a clause set is highly redundant (as C in fact is), the redundancy may be such that the use of models can ameliorate its effect on the inference space size.

Yet another reason for using models which has been often cited is that humans are ineffective theorem provers when they are without a model for the syntax they manipulate. This seems to be a valid point in general, but does not constitute support for the specific way that models are used in a resolution strategy such as TMS. It is likely that humans use models in ways that have not been discussed in this thesis.

4.6.2 Arguments for Model Schemes and Structures.

The main idea in using model schemes instead of single Herbrand interpretations is to have a theoretical framework adequate to handle the pragmatic situations where sophisticated models are to be employed, but where these models are not to be completely defined (either because they cannot be, or because they should not be, completely defined).

There is a fundamental incompatibility between the theory of semantic refinements and the pragmatics of executing theorem proving searches subject to those refinements. The theoretical statement of a semantic refinement and elucidation of its properties
(e.g. completeness) is best handled by the use of Herbrand interpretations. But a Herbrand interpretation potentially contains an infinite amount of information (albeit mostly irrelevant) and thus is in general impossible to embody in an actual implementation. The usual method, whereby a finite implementation uses a single Herbrand interpretation, is simply to use only those Herbrand interpretations which can be specified in some simple way. The homogeneous partition settings (Loveland, 1978) are examples of this type of specification of models.

A different way out of the specification problem is to only partially specify a Herbrand interpretation. Here a ground literal is said to be true if it is "in" the partial specification, and false if it is not in the partial specification. The settings which are not partitions (Loveland, 1978) are examples of this type of model specification.

The primary difficulty with these two ways of defining a model is that they lack the flexibility necessary to define the models which we really want to use in semantic refinements. The second approach given above, that of only giving a partial specification, is a step in the right direction, since it leaves open the option of simply not specifying certain information which is irrelevant. Such a partial specification of a Herbrand interpretation does define a model scheme. However "partial specification" of a
Herbrand interpretation is itself not enough. What is needed is a compact, flexible, and powerful way of representing sophisticated models, and which can be carried over into the pragmatics of evaluations of clauses in actual implementations.

The approach taken in this chapter is to theoretically define semantic functions based on model schemes as partially specified Herbrand interpretations. The Θ-functions are then the basis for showing how sound semantic functions can be defined. It is the model structure portion however which allows sophisticated model schemes to be easily specified, both theoretically, and in practice. The key to this is the change from specifying the truth values of individual literals in an interpretation, to instead giving the relationships between literals in the interpretation, in the form of the scheme defining set. For theoretical purposes the use of arbitrary sets of satisfiable first order sentences as the means of model specification is appealing and utilitarian. For most sophisticated models however, a procedure will be required for model evaluations. As in the case of \( \gamma^{\Theta_{\mu}} \) of section 4.5, the soundness of the semantic function determined by the procedure can be analyzed and supported by arguments based on model structures and model schemes.
4.6.3 The Role of False Permissiveness.

Strategies such as TMS, HLR and the semantic resolution of Slagle, are false permissive complete, and thus are complete when using any sound semantic function. This allows one to move away from using $\phi$ (which evaluates correctly with respect to $h$) to any $\psi$ which is false permissive with respect to some $h$ (for the appropriate language), that is, any sound semantic function, $\psi$. The overall view of models developed in sections 4.2 and 4.4 could not be accomplished without the freedom to allow increasing false permissiveness in the functions defined from one step to the next. In section 4.6.5 some specific comments will be made about Theorem 4.2-5 with respect to its contribution to false permissiveness.

Besides the above role of false permissiveness, there is its role in making some model evaluation processes workable in practice. For certain model structures the resulting class of questions (in the form of sets of literals to be evaluated) may be undecidable. If most sets of literals can be quickly evaluated by a known process, however, then simply assigning false to those sets which exceed certain effort limits may give overall a model which is adequate. The procedure $\phi_{SLC}$ is an example of this situation in a slightly different form, where a decidable domain (Ferrante and Rackoff, 1975) was implemented by a procedure which was incomplete, but still gave good
4.6.4 Generality of the Model Scheme Approach.

It should be stressed that
1. $\gamma^{B,c}$ is only one specific example of a sound semantic function based on a non-theorem proving evaluation mechanism (in this case simultaneous linear equations);
2. even in the case of just simultaneous linear equations, there is a very large freedom of choice of parameters in setting up the evaluation procedure.

With respect to the second item, several degrees of freedom were already quite apparent in Example IV. These were the ability to add equations to $C$, without destroying the soundness of $\psi^{\models C}$, the choice of $<_A$, and the freedom to have different choices of recognition capability in steps 3 and 4 of the specification of the algorithm $A$.

A less apparent but extremely important degree of freedom in defining $\beta_{c,e}$ is in how literals in $L_{\gamma\gamma}$ (translated by $T$ from literals in $L_c$) are themselves translated into equations. Because the clause set $C$ of Example IV was axiomatizing an ordered group, we chose to map $\text{EQ}$ (in $L_c$, corresponding to "$=\text{in } L_{\gamma\gamma}$) into the usual notion of equality in the procedure $\beta_{c,e}$. Thus $=\text{in } L_{\gamma\gamma}$ becomes the equation $x - y = 0$. But this is not necessary. For
example, a sound semantic function results from mapping \(=(x,y)\) to \(nx + my = 0\), where \(n\) and \(m\) are any integer constants. This will yield a model which does not correspond to the usual intended meaning of equality (assuming \(n \neq -m\)), but there is nothing to prevent doing this. The only restrictions are that it must be done uniformly, that is, once the mapping is chosen it may not be changed, and the mapping of \(\neg(x,y)\) must actually be correctly related to the mapping of \(=(x,y)\); in this case \(\neg=(x,y)\) must map to \(nx + my + \Delta r = 0\), for \(r\) a "new" integer. The reader should be able to see that instead of changing how \(\mathcal{G}_{L^k}\) interprets the standard symbols in \(L_{\mathcal{M}}\) of Example IV, the same effect could be achieved by suitable modifications of \(T\), \(L_{\mathcal{M}}\) and the procedure for forming the equations in \(L_{\mathcal{M}}^*\) from sentences in \(L_{\mathcal{M}}\).

Thus we see that \(\mathcal{G}_{L^k}\) is really an extensive family of models, and is not quite so simple as was illustrated in Example IV. We wish to further stress that although there exist decision procedures for consistency checking of the class of sets of equations on which the procedure \(A\) operates, Example IV defined \(A\) only loosely, specifically allowing it to be incomplete.

The procedure \(\mathcal{G}_{L^k}\) was chosen as Example IV because it is likely to be intuitive to the reader, since simultaneous linear equations are commonly understood. Clearly there are many mathematical systems which can be used in similar ways,
e.g. Boolean algebra, analytic geometry, non-linear equations over real numbers, and Presburger and other subtheories of arithmetic. The possibilities for procedures are enormous (and not restricted to be what are typically thought of as mathematical systems) having only two main criteria to satisfy:

1. There must exist some satisfiable axiomatization of the procedure, which justifies the procedure in the way \( \mathcal{E}_{\text{proc}} \) justifies \( \Theta_{\text{proc}} \).

2. The procedure must be pragmatically effective, i.e. detect most inconsistencies in relatively short time and always terminate reasonably quickly.

4.6.5 Problems for Further Investigation.

We now indicate some problems with the current view of models and areas that seem important for future research.

There is no firm knowledge, theoretical or empirical, pertaining to the problem of choosing a model which maximizes efficiency for a given clause set and refinement combination. It seems reasonable to assume (and there is some slight empirical evidence backing this up) that for most situations one of the better possible models is the human intuitively correct model. But there is no evidence to show that in general the very best models are not consistently the unintuitive models. It was argued in section 4.6.1 that if complexity is measured as number of
false clauses (using appropriate models) then as clause sets become rapidly more complex by other measures, the number of false clauses tends to remain small and provides a strong constraint on the number of level 1 clauses in a strategy such as TMS. This suggests that the best models for a given clause set are those which make as many clauses true as is possible, but does not firmly establish this to be so. Also, since often several distinct models can be found which make the same number of clauses true, there is a necessity for some other criteria for model selection. It would seem that a much more adequate understanding of this issue is mandatory if efficient progress is to be made in incorporating models into theorem proving.

Even assuming we knew some criteria that a model should meet in order to be efficient in guiding a search process for a particular input clause set and inference rule, we would still be left with the problem of constructing a model (i.e., a specification of a semantic function or evaluation process) which met the criteria. Clearly a mechanical procedure for model construction would be desirable. Very little work has been done in this area, but its potential impact seems large. If practical methods of model construction can be developed, it is possible that in certain situations it may be more efficient to spend rather large amounts of effort in searching for a particularly good model for a clause set, instead of spending the effort in generating clauses in Lc according to an inferior model. Of
course, the two alternative uses of search effort are not really mutually exclusive, and a back and forth dialogue between model construction and search space expansion would probably ultimately be desirable.

There is a technical deficiency with the view of models given in this chapter, which while it can be remedied, requires some investigation as to the best method (or methods, as several may be appropriate depending upon circumstances) for handling it. This deficiency is that an excessive amount of false permissiveness can arise for certain models because of the one way implication in Theorem 4.2-5. As an example of the difficulty, assume we wished to model a language, $L_C$:

$$\text{desc}(L_C) = \langle \{ P'(-) \}, \{ a', b' \} \rangle,$$

and it was decided that the model should simulate the Herbrand interpretation for $L_C$

$$h = \{ P'(a'), P'(b') \}.$$ 

If we try the obvious, namely:

$$\text{desc}(L_M) = \langle \{ P(-) \}, \{ a, b \} \rangle$$

$$\mathcal{T}(x') = x \text{ for } x' = P', a' \text{ or } b'$$

$$\mathcal{T}(x) = x \text{ for all other } x$$

$$\Sigma = \{ P(a); \ P(b); \}$$

then the expectation is that this model structure will have $\gamma^x = \phi_h$ (with $\gamma^x$ from Theorem 4.2-8). The reason for expecting this is that
\[ \mathcal{M}_\Sigma^{L_{\Sigma}} = \mathcal{M}_\Sigma^{L_{\Sigma}} = \{ (P(a), P(b)) \} \]

\[ \mathcal{M}_\Sigma^{L_{\Sigma}} = \{ (P'(c'), P'(b')) \} \]

i.e., all the model schemes are singleton classes. However, \( \phi_A(P'(x)) = \text{true} \), but \( \psi_\Sigma(P'(x)) = \text{false} \), since

\[ \mathcal{T}(\neg\{P'(x)\}) \cup S = \{ \neg P(c) \} \cup S \]

is satisfiable, where \( c \) is a new Skolem constant. The discrepancy arises because the function \( \phi_A \), by its definition, looks just at the ground instances of its argument over the Herbrand universe, while \( \psi_\Sigma \), by trying to operate through the notion of satisfiability, does not have any way to restrict "c" to be equivalent to "a" or "b" in \( L_{\Sigma} \). The problem is easily fixed in this case, since the Herbrand universe is finite, by adding to \( S \) the following axioms:

\[ = (x, x), = (x, b); \]

\[ \neg P(x), \neg (x, y), P(y); \]

along with the reflexivity, symmetry and transitivity axioms for "=". For situations where the Herbrand universe is infinite however, no satisfactory general solution to the problem has been found (allowing infinite deductions is not considered an acceptable solution). That this problem does not always arise can be seen from the simultaneous linear equation model of Example IV.

Another area of investigation is that of replacing the notion of a scheme defining set, \( \Sigma \), as a satisfiable set of first order formula, with that of \( S \) being a set of formula in some other logic. If we are still doing first
order resolution in $L_\xi$, then the question arises as to whether or not a higher order specification of $\Sigma$ can lead to a semantic function which has more desirable properties.

This concludes the discussion of our view of models. Some further perspectives relating to models as used specifically in HLR are in the next chapter.
Chapter 5

DISCUSSION OF HLR

In this chapter we discuss some characteristics of the HLR refinement, concentrating on the relationship between HLR and the models used, as well as pointing out some areas for future research. Among the issues to be treated are the reasons for an expected efficiency gain (in reduced number of clauses), the increased overhead (for FSL's) in HLR, and the compatibility of HLR with other strategies.

5.1 Refinement Strength of HLR

From the presentation in this thesis it should be clear that HLR is a refinement of TMS + FSL, and TMS + FSL is a refinement of TMS (when using the same Herbrand semantic function in each case). Also Figures 1--1 and 2--3 show that TMS + FSL is deduction incomplete with respect to TMS, and Figures 2--3 and 2--1 show that HLR is deduction incomplete with respect to TMS + FSL. That this will translate into an efficiency increase in the overall theorem proving process can only be determined by experimentation, unfortunately, since HLR is, like most other resolution refinements, resistant to theoretical efficiency analysis. It has been argued that a certain type of deduction incompleteness is desirable for efficiency (Minicozzi and Reiter, 1972) (Wos and Robinson, 1973) but the connection is tentative. Some of the factors involved in efficiency can
be identified however, and will be treated in this section.

It is clear that some type of ordering of literals in clauses is a necessity in resolution. Without it the conjunctive normal form combined with the resolution inference rule embodies a tremendous redundancy. That LR is the proper direction to take in controlling this redundancy is intuitively plausible, but until adequate (empirical or theoretical) evidence for this can be found, it will be taken as a working hypothesis. Extension of LR to HLR by adding double lock numbers constitutes a weakening of the literal selectivity of LR, but (again intuitively) the ability to incorporate the focus contributed by TMS which this allows seems to be a reasonable direction to take.

Empirically HLR has performed quite well at the ground level (i.e. the sentential calculus) on a variety of realistic size clause sets and models. In these cases HLR is simply LR with an HL-proper lock numbering (using single lock numbers). These experiments cannot positively prove the reasonableness of using LR and TMS together at the first order level proper, but at least give support to it, since no deleterious effects of imposing HL-proper lock numberings, vis a vis other numberings, was observed at the ground level.

At the first order level proper however, the primary concern is the control of what terms are being formed in the search space. It is here that HLR has its greatest strength
as a refinement, i.e. in blocking unnecessary deductions. This arises by the accumulation of FSL literals, which act as constraints on how the clauses can be used as parents. Notice that TMS + FSL is not able to build FSL's as restrictive as HLR, since, as can be seen by the definition of true factoring, HLR can move certain literals into the FSL of a true factor just on the basis of lock numbers alone.

From a different perspective we can say that HLR is a strong refinement because it has a property we call ground faithfulness. In order to simplify the notion of ground faithfulness we will consider sets of literals to be multi-sets (i.e. a collection of occurrences of literals where the same literal may have multiple occurrences). The inference rules for clauses as multi-sets, as opposed to sets, will not be given explicitly, but are to be the intuitively natural and obvious modifications of the inference rules for ordinary sets of literals. A refinement, R, is said to be ground faithful iff for all clause sets, S, and all R-deductions, D, from S, there exists at least one ground deduction, Dφ, such that:

1. Dφ is an R-deduction from Sφ;
2. Sφ is a set of ground instances of clauses in S;
3. D is a "lifted" version of Dφ (in the obvious sense of D and Dφ being tree isomorphic, and each node of Dφ is a ground instance of the corresponding node in D).
Conditions ii and iii and the statement "$D_\phi$ is an unrestricted resolution deduction from $S_\phi$" are automatically satisfied for any refinement of resolution on multi-sets. But for some refinements on multi-sets there exist r-deductions $D$ at the general level for which no $D_\phi$ satisfying ii and iii is also an r-deduction. It is obvious that certain refinements are ground faithful, e.g. unit resolution or set of support (where in the ground set $S_\phi$ a clause is supported if it is an instance of a supported clause in $S$). The presentation in Chapter 2 of this thesis shows TMS (using a Herbrand semantic function) not to be ground faithful. Examples exist showing that semantic resolution (Slagle, 1967) is not ground faithful (again using a Herbrand interpretation as the model).

We claim, without presenting proof, that TMS + FSL and HLR are both ground faithful when the semantic function is that for any Herbrand interpretation for any language of the clauses. To prove ground faithfulness for these strategies is tedious and not appropriate here. Counterexamples exist to show that TMS + FSL and HLR are not ground faithful when an arbitrary sound semantic function is used.

It is fair to say that a primary motivation for and much of the utility of the FSL notion is centered on ensuring ground faithfulness. However, in HLR using an arbitrary sound semantic function, where ground faithfulness does not necessarily hold, there is another recognizable
effect from the FSL mechanism. We call this effect model
scheme splitting, and the remainder of this section is a
discussion of this effect.

From the proof of Theorem 3.1-1 (False Permissive
Completeness of HLR) it is clear that if h is a Herbrand
interpretation for the language of the clauses, and \( \psi \) false
permissive wrt. \( \phi_h \), then HLR using \( \phi_h \) is a refinement of
HLR using \( \psi \). To show that, in general, HLR using \( \psi \) is not
a refinement of HLR using \( \phi_h \) is easily done by just
considering a few example clause sets and taking \( \psi \) as the
function mapping all sets of literals to false. Thus in
general the false permissiveness of \( \psi \) wrt. \( \phi_h \) can allow
extra clauses in the inference space.

Now consider a practical theorem proving situation
where a sophisticated model has been set up, and let \( \psi \) be
its semantic function (e.g. \( \psi^{HSL} \) of Example IV). We assume
\( \psi \) is sound, but that it is not identical to \( \phi_h \) for any h a
Herbrand interpretation for the language of the clauses.
What is of concern is that we know a priori that to whatever
degree \( \psi \) is giving "extra" false evaluations of sets of
literals there will be "extra" clauses in the search space.
The question is: How sensitive is the hastening of the
combinatoric explosion of the search space (due to false
permissiveness of \( \psi \)) to the amount of false permissiveness
of \( \psi \)? Clearly this is a question which is not precisely
formulated, but it is unlikely to be answered adequately in
a quantitative sense by abstract analysis, and so we leave it as an ill-defined qualitative question.

Model scheme splitting is an effect which has the potential for reducing the effects of extra false answers from a sound semantic function. In the discussion to follow we assume that \( \Psi \) not only is sound, but exhibits regularities of behavior which are likely to hold in practical situations, namely:

i. \( \Psi \) evaluates a set of literals containing a complementary pair as true;

ii. if \( \Psi(K) = \text{true} \) and \( K \subseteq K' \) then \( \Psi(K') = \text{true} \).

There are two ways to consider intuitively what an FSL does for a clause with respect to model schemes. The first is to say that within a given Herbrand interpretation in a model scheme, the FSL restricts the class of ground instances of the standard literals which we consider when evaluating the t.v.d. of the clause. This can be seen from the definitions of the t.v.d. function and the \( \xi \)- and \( \phi \)-statements. The second way to view the FSL is to say that it defines a subset of a model scheme in which the standard literals of the clause are allowed to represent some ground instances. It is this second view of FSL's which we wish to emphasize when considering model scheme splitting.
Consider two clauses
\[ \mathcal{H}' = \langle S(\mathcal{H}'), \{ P(a,b,c) \} \rangle \]
\[ \mathcal{H}^2 = \langle S(\mathcal{H}^2), \{ \neg P(a,b,c) \} \rangle, \]
and the semantic function \( \psi^{\mathcal{U}}_{\mathcal{M}_2} \) of section 4.5, with the empty context set. Both of these clauses are \( \mathcal{C}^{\mathcal{U}}_{\mathcal{M}_2} \)-feasible, since with an empty context set there is no way inside the model to decide whether \( a + b = c \) or not.

Suppose \( \mathcal{H}' \) and \( \mathcal{H}^2 \) can be resolved to yield \( \mathcal{H}^3 \):
\[ \mathcal{H}^3 = \langle S(\mathcal{H}^3), \{ P(a,b,c), \neg P(a,b,c) \} \rangle \]

(N.B. For simplicity in this example we assume that any standard literals of \( \mathcal{H}' \) and \( \mathcal{H}^2 \) that are to go into \( S(\mathcal{H}^3) \) are already in \( S(\mathcal{H}') \cup S(\mathcal{H}^2) \).) Now \( \mathcal{H}^3 \) is infeasible because
\[ \{ P(a,b,c), \neg P(a,b,c) \} \]
translates into an inconsistent set of simultaneous linear equations, and we assume the inconsistency is recognized.

In this case the infeasibility of \( \mathcal{H}^3 \) does not arise from any resolution unifier being applied to FSL literals of its parents, nor from any standard literals of its parents having descendants in the FSL of \( \mathcal{H}^3 \). Rather \( \mathcal{H}^2 \) is infeasible because its two parents were feasible only over disjoint subsets of some underlying model scheme. Thus \( \psi^{\mathcal{U}}_{\mathcal{M}_2} \) in effect says that \( \mathcal{H}' \) and \( \mathcal{H}^2 \) are separately legitimate members of the inference space, but they cannot interact (and neither can any of their descendants) because they belong to different model scheme subsets. This is the effect we call model scheme splitting.
To gain a clearer feeling for this effect in the context of false permissive functions, notice that there can be no Herbrand interpretation, \( \mathbf{h} \), such that \( \phi_{\mathbf{h}} \) will behave as \( \gamma_{\phi_{\mathbf{h}}}^{\Omega} \) for the \( \mathcal{H}^1, \mathcal{H}^2 \) and \( \mathcal{H}^3 \) above, i.e. make \( \mathcal{H}^1 \) and \( \mathcal{H}^2 \) feasible, but \( \mathcal{H}^3 \) infeasible. The reason is that the FSL literals of \( \mathcal{H}^1 \) and \( \mathcal{H}^2 \) are ground and exact negations of each other, so that \( \phi_{\mathbf{h}} \) must judge exactly one of them as true, and therefore that clause is infeasible.

What this means for search space combinatorics is that while \( \gamma_{\phi_{\mathbf{h}}}^{\Omega} \) admits "extra" clauses in the search space (relative to some \( \phi_{\mathbf{h}} \) such that \( \gamma_{\phi_{\mathbf{h}}}^{\Omega} \) false permissive wrt. \( \phi_{\mathbf{h}} \)), these clauses tend to have limited interaction with each other. Thus there is reasonable hope that the FSL notion will nicely control the effects of false permissiveness of the sound semantic functions which are likely to be used in practice.

For the reader who thinks that the problem of false permissiveness of some \( \gamma \) can be handled by just replacing \( \gamma \) by some \( \phi_{\mathbf{h}} \) such that \( \gamma \) is false permissive wrt. \( \phi_{\mathbf{h}} \), should note that the assumption is that \( \gamma \) is implementable and known, while any such \( \mathbf{h} \) may not be tractable in a practical situation, and possibly is not even effectively computable from \( \gamma \).
5.2 Overhead of HLR.

The additional overhead in HLR of maintaining double lock numbers is completely negligible in comparison to the work load in any resolution system for computation and storage of resolvents.

The increased overhead of storing and evaluating FSL sets is likely to be significant. There are two competing desiderata for FSL's. The first is that one would like FSL sets to be as restrictive as possible, as this would control the fecundity of clauses. The second is that, at some point, the overhead to evaluate very large FSL's in clauses, for feasibility and t.v.d. evaluations, becomes prohibitive.

It is easy to see, by a crude approximation, that the FSL size of clauses tends to grow as $2^k$, where $k$ is the depth of the clause, and clauses at level 1 will have FSL set sizes on the order of the average size of the standard literal sets of an input clause. Under such growth behavior searches extending tens of levels deep would be impractical in general. Storage of FSL's is not particularly a problem, since the structure sharing scheme (Boyer and Moore, 1972) can easily be extended to cover the FSL literals. The evaluation effort (time) for very large FSL sets (hundreds or thousands of literals) would in general be prohibitive, although in some special cases of models this may not be so.
For reasonably sophisticated and realistic models one would expect that only clauses with a relatively small FSL set will be tractable, certainly less than 100 literals. Thus if search spaces are to be expanded tens of levels deep, some mechanism for reducing the size of FSL sets must be employed. The least refined such mechanism is simply to delete FSL literals until the FSL set is some acceptable size. Which literals to delete and what size is acceptable might be determined by heuristic criteria. Note that if on the average for each inference step, roughly 50% of the FSL literals in the resolvent can be eliminated then FSL size will be roughly constant. This applies in both the case of simple deletion and for the discussion below.

In some realistic problems there is reason to believe there will be a sufficiently strong mediating effect, which we call constraint saturation, which will allow the FSL's to be kept acceptably small while still retaining all possible information. Constraint saturation refers to the situation where, when the standard literal set of a clause is small, and the FSL is large, it is exceedingly unlikely that most of the FSL literals contain information independent of the information contained in the other literals. A considerable amount of constraint saturation has already been experimentally observed in an HLR implementation working on group theory problems. It is unknown just how well FSL size could be controlled, however, since this implementation had no capability of recognizing and eliminating the redundancy
among the FSL literals. As an example of the type of redundancy among FSL literals which can occur, consider the clause

\[ Q_1 = \langle S(Q_1), \{ \neg P(x,y,c), \neg P(c,g(x),y) \} \rangle. \]

If we use the semantic function \( \psi^{FSL}_{Q_1} \) of Example IV then the procedure translates both literals in \( \Phi(Q_1) \) into the same identical equation. Thus one of these literals can be simply deleted from the FSL of \( Q_1 \). It is easy to see that deleting one of these literals will not change the search space in any way other than reducing the number of FSL literals in \( Q_1 \) and its descendants. There are many other cases where FSL literals can be eliminated. For example if

\[ Q_2 = \langle S(Q_2), \{ \neg P(x,1,x), \ldots \} \rangle \]

then the explicitly written FSL literal may be deleted from the FSL set of \( Q_2 \) if the same semantic function, \( \psi^{FSL}_{Q_1} \), is being used. Care must be used in deciding when an FSL literal is actually redundant or extraneous however, if one does not want to lose any relevant information. For example, just because an FSL literal is a ground literal does not mean it is redundant, as is seen for the example in section 5.1 illustrating model scheme splitting. We mention that the ability to eliminate literals from the FSL without changing the context of the inference space depends upon the procedure used in the semantic function evaluation as well as the literals themselves. For example, if the procedure has an effort cutoff, then a particular FSL might be evaluated to false simply because the evaluation process
stops before it is proven that it is true. Yet some subset of the same FSL (e.g. arising from eliminating some redundant literals) might be evaluated to true because the effort cutoff is larger than the effort required for the smaller set of literals. Thus a clause could be feasible, and its simplified form infeasible. Such an occurrence cannot happen with a Herbrand semantic function, since by their nature the Herbrand semantic functions are not dependent upon such "non-logical" attributes of FSL's as number of literals, nor dependent upon the degree of redundancy, etc. of the FSL literals.

We will not attempt to characterize the possibilities for FSL simplification further, since more empirical work is necessary before this is profitable. The issue of reduction of FSL size is a complex one. The need to control the size of FSL sets is oriented more toward reducing the effort required to evaluate them, rather than reducing the storage requirements. As such it is not clear if minimal effort will generally occur with smallest FSL size, or whether some redundancy is desirable. In this matter there is likely to be no uniform answer, but rather different answers for different languages and model procedures.

Throughout this thesis FSL literals have been treated as syntactic literals in the language of the clauses. In an actual implementation of HLR however, it may be more convenient to store FSL's in terms of their model
representations. In certain cases this may facilitate the reduction of the redundancy in the FSL. For example, when using the simultaneous linear equation model used in Example IV, if each FSL is stored as a set of equations in Hermite normal form, then in effect many redundant literals, giving rise to redundant equations, could be eliminated from the FSL. The potential for reducing the number of equations is large, but the price to be paid is that structure sharing is no longer applicable (at least not in the simple way it would be if no reductions were being done).

Clearly there is need for experimentation to establish workable configurations for FSL storage so that both the amount of storage and the evaluation effort are within acceptable limits.

5.3 Experimental Results -- Domain of Applicability.

A rudimentary HLR implementation has been used in experiments with clause sets covering a wide range of problem difficulty. The only models available were the simultaneous linear equation (SLE) model and the totally trivial models (deciding truth values only on the basis of the predicate letter and sign of the literal). The HLR implementation differed somewhat from the detailed prescription given in this thesis in order to simplify both the task of implementation and the effort required for model evaluations during execution. The resulting inference space
of the implementation maps into the inference space of HLR as described in this thesis in a simple way, so that the empirical results obtained are directly applicable (the implementation used the primitive representation form for HLR, described in (Sandford, 1977a)).

HLR tends to find deep and narrow refutations in comparison to most other refinements. This is due to the LK component. With certain realistic clause sets (group theory and other algebraic type problems) using the SLE model, the first few levels of the inference space are exceedingly thin (input sets of 10 to 20 clauses yielding less than 10 new clauses at each of the first 3 or 4 levels). After the first few levels the rate of search space growth increases rapidly. The implementation used breadth first search, with no simplification or deletion strategies, and no cutoffs of any type, such as term complexity or number of literals. Even so it was possible to handle some of the easier clause sets that have often been used for theorem proving examples in the literature. In order to handle some of the more difficult clause sets several additional strategies, which we will refer to as auxiliary strategies, had to be added to the implementation. Only a very few clause sets were tried using the auxiliary strategies (the implementation became impractical because of the large amount of effort required to set up the test runs when the auxiliary strategies were used) but several relatively difficult problems were able to be handled in this mode.
The auxiliary strategies will not be described in detail, but briefly they are:

1. an extension of a subset of the schemata of (Overbeek et al., 1976), which partially compensates for the lack of explicit subsumption and tautology deletion in the implementation;

2. a trivial form of demodulation, where one constant symbol may be substituted for another;

3. a breadth first search, which, whenever a new unit clause "=(c1,c2); FSL = { . . . }" is derived with c1 and c2 constants, substitutes c1 for c2 in the input clauses, deletes the remainder of the search space, and re-starts the search space expansion from the new input set.

An interesting effect occurred with respect to the auxiliary strategies. Before using the auxiliary strategies a large amount of experimentation was done, and only a weak superiority of the SLE model over the trivial models was found (restricting attention only to those clause sets where the SLE model was intuitively the correct model). After adding the auxiliary strategies, for a few of the same clause sets (only a few were tested) the SLE model improved its performance much more than did the trivial models. This indicates that the types of inefficiencies handled by the auxiliary strategies were somehow mostly circumvented already by the trivial models. It is confounding effects such as this that make it extremely difficult to evaluate resolution strategies empirically. In general the
efficiency of HLR vis a vis other strategies was favorable to HLR (although no linear refinements were tested in the comparison). On very simple problems (proof depths less than level 4 under unrestricted resolution, and 10 or fewer input clauses) the trivial models generally worked better than the SLE model, even when the SLE model was intuitively appropriate (N.B. with the trivial models HLR is simply LR with an HLP-proper single lock numbering). On more difficult problems the SLE model was superior to the trivial models. For clause sets where the SLE model is intuitively appropriate the changeover point was roughly 10 clauses in the input set and a proof depth of level 4 (this is without the auxiliary strategies). The reasons for this behavior are not well understood at this time.

5.4 Compatibility With Other Strategies

5.4.1 Refinements.

A very nice feature of HLR is its potential refinability with respect to its LR component. By this is meant that if R is a refinement of LR which is complete for HL-proper (single) lock numberings at the ground level, then R can be combined with THS to give a complete general level refinement of HLR. The justification for this statement is Theorems 2.2-1 and 2.2-2, where the substitution of R for LR leaves the theorems and proofs still true, thus giving the basis for assigning double lock numbers to literals at the
general level. Theorems 2.3-1 and 2.3-2 then provide the intuitive basis (but not the proof) that FSL's can be added to a general level version of R using HL-proper (double) lock numbers. Unfortunately LR is not compatible with very many of the known refinements of resolution. In (Boyer, 1971) LR is shown by examples to be incomplete when used in conjunction with some other well known refinements.

The prospects for further improvement of HLR are not completely barren however. In (Sandford, 1979) a refinement strategy is presented which is an outgrowth of the schemata mechanism in (Overbeek et al., 1976) which is complete when used in conjunction with HLR. In addition it appears that certain variations of LR which are not pure refinements of LR are also able to generate variations of HLR.

Loveland has pointed out (personal communication) that HLR is a special case of a more general refinement in which the literals in a clause which act as true literals can be ordered by ordering rules more general than the ordering determined by inherited lock numbers. It is possible to generalize the $Q^T_A$-deduction given in (Loveland, 1978) to non-homogeneous settings, and then consider HLR as a special case of this in which the true literals are ordered by inherited lock numbers. There seems however to be little reason to do this when considering the nature of the possible changes in the inference space. The reason is that the following property holds for HLR when using a Herbrand
semantic function, and when the initial HL-clauses all have distinct lock numbers on their literals:

Suppose \( H \) is a derived (feasible) clause, and \( K \) is a subset of \( \mathcal{S}(H) \) such that each literal in \( K \) has the same double lock numbers as every other literal in \( K \), then \( K - \text{FSL}(H) \) is either empty or is a singleton set (we assume no FSL simplifications have been done).

Under such circumstances the utility of an ordering for true literals determined by some criteria other than inherited lock numbers would seem to be (in most realistic clause sets) of relatively little importance in controlling search space growth rates.

When using a semantic function which is not a Herbrand semantic function, then there is a possibility of several "true" literals in a clause having the same lock numbers. However, for this to happen the semantic function must exhibit such behavior that its overall utility would be in question, specifically (if no FSL simplifications have been done) the semantic function must evaluate at least some sets of literals as false which contain a complementary pair of literals.

In the most general case, where an arbitrary semantic function is being used along with FSL simplifications, then some form of marking of literals in resolvents could be done, so that all of the standard literals from the false parent are marked, and otherwise all literals in resolvents have the mark status of their parent literals (a merge literal is marked if either parent literal is marked). All
marked literals are to be considered as false literals only. Doing this will ensure that no two standard literals with the same lock numbers can be selected true literals for factoring. If the semantic function is sound, then such a marking restriction will not destroy completeness.

Finally, it is easy to show that factoring can be restricted so that only literals with the same lock numbers are factored together. In view of the discussion immediately above, this then implies that in definition 2.4.20 (true parent factoring) the set \( u \) of \( c_4, c_5 \) and \( c_6 \) can be restricted to be a singleton set, specifically \( u = \{ e_6 \} \).

5.4.2 Simplification Strategies.

Lock resolution is known (Boyer, 1971) to be incomplete when combined with the usual elimination of tautologies (lock numbers are invisible when deciding if a clause is a tautology). This fact is known by an example given by Boyer. However this example does not have an HL-proper lock numbering (for any Herbrand interpretation). Thus it is still an open question as to whether or not LR is complete with tautology elimination when the lock numbering is HL-proper. HL-resolution will be complete with tautology elimination iff LR with HL-proper (single) lock numberings is complete with tautology elimination.
The completeness of subsumption has not been investigated in the context of HLR. It has not been determined yet if the recent subsumption results of (Loveland, 1978) for indexing refinements are directly applicable to HLR, even when using a Herbrand semantic function.

5.4.3 Ordering Strategies.

The use of (search control) ordering strategies is poorly understood in general in resolution, and it is difficult to establish any strong prescriptions at this time for HLR. It would seem to be the case however that heuristic ordering functions that weight clauses on the simple basis of length and term complexity are likely to be less effective in HLR than they are in most other resolution inference rules. This is mainly because of the LR component of HLR, which often admits only simplest proofs which are significantly more complex (measured by term height or clause length) than other strategies. On the other hand there is already available (in the FSL’s) in HLR much more information concerning term structure than in most other strategies. This may make it possible to devise a more sophisticated ordering mechanism based on term structure rather than just simplest terms first. We point out here that TMS by itself also can cause increases in maximum term height and maximum clause length, necessary for a refutation, over that in unrestricted resolution. In
(Luckham, 1970) it is incorrectly asserted that term height does not increase using TMS in an actual search space expansion, and therefore does not require modification of editing strategies based on term height. Very simple examples exist to show that TMS indeed can increase required term height.

5.5 Areas for Future Research.

The limited experimental work that has been done with HLR in combination with the auxiliary strategies indicated that, when sophisticated models are to be used, HLR suffers strongly from the lack of subsumption. Thus this is an important area for both theoretical and empirical work. Tautology elimination is also an appropriate area of investigation. It is not yet known if the RL-proper lock numbering allows stronger results to hold than for arbitrary lock numberings.

As mentioned previously, strategies for reducing the size of FSL sets are needed which will allow both rapid feasibility and t.v.d. evaluations to be made while avoiding loss of significant constraining power of the FSL's by over simplification. Since it seems to be the case that the appropriate techniques will depend upon the type of model used, it is also important to gain experience with a variety of sophisticated models.
Building in special theories such as equality into HLR should not offer any unusual difficulties, but no attempt has been made in this direction. The approach used in (Brand, 1975) may be of utility in this connection.

Perhaps the greatest potential for HLR lies in its well integrated use of models. We have here in mind both additional semantic restrictions of the refinement type and sophisticated search control strategies based on semantic information. The initial research that led to the material in this thesis came out of an effort to develop the semantic framework to hypothesize (by automatic procedures) high level proof outlines, or schema, which would guide a resolution search process. Such approaches to theorem proving are mandatory when the theorem complexity is much higher than that typically attempted with theorem provers today. One of the primary difficulties of such an approach is establishing a mechanism of efficient feedback so that an initial proof outline which proves inadequate can be the basis for hypothesizing a new proof outline which avoids the errors of the old one. While no substantial progress has yet been made in this area of theorem proving, it would seem that notions such as FSL's, which are both semantic and global over deductions, would play an important role.
BIBLIOGRAPHY

References preceded by "+" are not specifically cited in the text but generally relate to the thesis and are not in the bibliography of (Change and Lee, 1973).


of Computer Science, Rutgers University, 1977b.


# TABLE OF SYMBOL CONVENTIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Identification</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>algorithm for $\Phi_{IL}$</td>
<td>178</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>statements for t.v.d. evaluations</td>
<td>63</td>
</tr>
<tr>
<td>C</td>
<td>sets of literals, context sets</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>context set for $\Phi_{IL}$</td>
<td>174</td>
</tr>
<tr>
<td>$G$</td>
<td>sequence of context sets</td>
<td>162</td>
</tr>
<tr>
<td>D</td>
<td>deductions</td>
<td>--</td>
</tr>
<tr>
<td>$f$</td>
<td>truth value of $\Theta$-functions, selector function for false lock numbers</td>
<td>125</td>
</tr>
<tr>
<td>$F$</td>
<td>t.v.d. value</td>
<td>64</td>
</tr>
<tr>
<td>$\tau$</td>
<td>selector function for FSL literals</td>
<td>55</td>
</tr>
<tr>
<td>$G$</td>
<td>clause sets</td>
<td>--</td>
</tr>
<tr>
<td>$H, H, \tau$</td>
<td>HL-literals, clauses, clause sets</td>
<td>54</td>
</tr>
<tr>
<td>$h$</td>
<td>Herbrand interpretations</td>
<td>8</td>
</tr>
<tr>
<td>I</td>
<td>index sets</td>
<td>155</td>
</tr>
<tr>
<td>J</td>
<td>$\Sigma$-justification statement</td>
<td>152</td>
</tr>
<tr>
<td>K</td>
<td>sets of literals (HL- and normal), other sets</td>
<td>--</td>
</tr>
<tr>
<td>L</td>
<td>languages</td>
<td>--</td>
</tr>
<tr>
<td>$l$</td>
<td>literal core function</td>
<td>54</td>
</tr>
<tr>
<td>$M$</td>
<td>models, model schemes</td>
<td>122</td>
</tr>
<tr>
<td>$M$</td>
<td>model structures</td>
<td>123</td>
</tr>
<tr>
<td>Symbol</td>
<td>Identification</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>N, N′</td>
<td>normal literals, clauses, clause sets</td>
<td>54</td>
</tr>
<tr>
<td>P, (P_{slc})</td>
<td>procedures (for simultaneous linear equations)</td>
<td>152, 174</td>
</tr>
<tr>
<td>Q</td>
<td>example clauses</td>
<td>--</td>
</tr>
<tr>
<td>R</td>
<td>refutations, refinements, resolvents</td>
<td>--</td>
</tr>
<tr>
<td>R</td>
<td>refutation procedures</td>
<td>149</td>
</tr>
<tr>
<td>S</td>
<td>selector function for standard literals</td>
<td>55</td>
</tr>
<tr>
<td>s</td>
<td>clause sets</td>
<td>--</td>
</tr>
<tr>
<td>t</td>
<td>sentences of a language</td>
<td>--</td>
</tr>
<tr>
<td>t</td>
<td>truth value of $\Theta$-functions, selector function for true lock numbers</td>
<td>125, 54</td>
</tr>
<tr>
<td>T</td>
<td>t.v.d. value</td>
<td>64</td>
</tr>
<tr>
<td>T/F</td>
<td>t.v.d. value</td>
<td>64</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>translation function</td>
<td>121</td>
</tr>
<tr>
<td>t.v.d.</td>
<td>truth value designator</td>
<td>64</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>functions for model scheme evaluations</td>
<td>125</td>
</tr>
<tr>
<td>$\Xi, \eta, \iota$</td>
<td>deduction trees, inference steps</td>
<td>--</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>scheme defining sets</td>
<td>123</td>
</tr>
<tr>
<td>$\Sigma'$</td>
<td>sequences of scheme defining sets, overlapping class of schemes</td>
<td>161, 155</td>
</tr>
<tr>
<td>$\sigma, \varphi, \epsilon, \psi$</td>
<td>substitutions</td>
<td>56</td>
</tr>
<tr>
<td>$\nu$</td>
<td>semantic functions</td>
<td>58</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Herbrand semantic function for $h$</td>
<td>58</td>
</tr>
<tr>
<td>$\Delta, \Gamma$</td>
<td>projector functions for languages</td>
<td>119</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Symbol</th>
<th>Identification</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\chi_{\Omega}$</td>
<td>set of simultaneous linear equations corresponding to set of literals $\chi$</td>
<td>176</td>
</tr>
<tr>
<td>$\varphi_{\chi'/\chi}(X)$</td>
<td>result of &quot;resolving out&quot; $P'$ by use of $\varphi$, and $\varphi_{\chi}$</td>
<td>169</td>
</tr>
<tr>
<td>$\mathcal{M}_X$</td>
<td>model scheme in language $X$ induced by $\chi'$ in $L_m$</td>
<td>124</td>
</tr>
<tr>
<td>$\forall^\chi$</td>
<td>list of universal quantifiers for all variables appearing in ${}$</td>
<td>63</td>
</tr>
</tbody>
</table>

**Special Forms**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Identification</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sim$</td>
<td>negation sign, negation operator</td>
<td>54</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>empty conjunction</td>
<td>63</td>
</tr>
<tr>
<td>$\square$</td>
<td>empty disjunction</td>
<td>--</td>
</tr>
<tr>
<td>$\ast$</td>
<td>HL-null clauses</td>
<td>--</td>
</tr>
<tr>
<td>$</td>
<td>---</td>
<td>-----$</td>
</tr>
<tr>
<td>$\longrightarrow$, $</td>
<td>$-</td>
<td>logical consequence</td>
</tr>
<tr>
<td>$\Rightarrow$</td>
<td>correspondence between language and interpretation domain, &quot;is interpreted as&quot;</td>
<td>--</td>
</tr>
<tr>
<td>$\ast$</td>
<td>defined as, or stands for</td>
<td>--</td>
</tr>
</tbody>
</table>
INDEX OF DEFINITIONS

A (algorithm for $\Theta_{SLF}$) 178
$\alpha$-statement 63
against 66
alphabet (of a language) 120
amicable 120, 121
arity 120
associated semantic function 58

Binary HLR 66
$\Theta$-statement 63

class constants 175
clause sets (HL-) 54
clauses (HL-) 55
complete 8
constraint saturation 216
context set 162, 174
deduction incomplete 9
desc 119
equality of HL-literals 54
excess literal parameter 87

f 54, 125
F 64
false 58
false lock number 54
false lock number selector function 54
false parent 66
false parent factoring 68
false permissive 61
false permissive complete 83
false selected literal 66
false substitution list (FSL) 55
feasible 62

ground 57
ground faithful 209
grounds 57

H-augmentation 84
Herbrand base 8
Herbrand interpretation 8
Herbrand semantic function 58ff.
Herbrand universe 8
HL-clause 55
HL-deduction 75
HL-literal 54
HL-null clause 64
HL-proper (lock numberings)
single 35
double 65

HLR 73
HLR - t.v.d. 103
HL-refutation 75
HL-resolution inference rule 73
HL-set for 65

I-indexed semantic function 155
immediately deducible 75, 68
   Binary HLR 66
   HLR 73

inference rule
   Binary HLR 66
   HLR 73
   HLR - t.v.d. 103
   LR n TMS 34
   TMS 9
   TMS + FSL 43
   unrestricted resolution 3

inference step 11
input set 8

language (first order) 55, 119
   of the clauses 118, 123
   of the model 118, 123
   reduction of L2n 124

literal (HL-) 54
literal core 54
literal core projector 54
lock resolution 34
LR n TMS 34

mgu 57
minimally unsatisfiable 9

model
   scheme 122, 124
   scheme splitting 211ff.
   structure 123

normal clause (literal) 54
null clause (HL-) 64

overlapping class of schemes 155

partial semantic function 134
procedure
   sound and terminating 150
refinement  3
refutation complete  8
relational structure induced by h  60

scheme defining set (of sentences)  123
scheme splitting  211ff.
selected literals  73
selected literal for factoring  69,  71
self resolution  73
semantic function  58
semantic P-deduction  10
   with FSL's  51

semantic resolution  11
Σ-justification statement  152
sound
   semantic function  61,  134
   and terminating refutation procedure  150

standard literal set  55
step (inference)  11
strengthening sequence
   of context sets  162
   of scheme defining sets  161

strict binary tree  11
substitutions  56

t  54,  125
T  64
T/F  64
terminating inconsistency detector  152
Ο-functions  125
TMS  9
TMS + FSL  43
translation function  121,  123
true  58
true lock number  54
true lock number selector function  54
true parent  66
true parent factoring  70
true selected literal  66
truth value designator (t.v.d.)  64

unconstrained  55
unifiable  57
unrestricted resolution  3
unsatisfiable  62
Vita

David Michael Sandford

1945 Born September 12, in Totowa Borough, New Jersey, to James and Louise Sandford.
1963 Graduated from DePaul High School, Wayne, New Jersey.
1963-64 Attended Marist College, Poughkeepsie, New York.
1964-67 Attended Stevens Institute of Technology, Hoboken, New Jersey.
1967 B.S. in Physics, Stevens.
1967-69 Active Service, United States Army.
1969-72 Graduate work in Physics, Rutgers University, New Brunswick, New Jersey.
1972 M.S. in Physics, Rutgers.
1972-79 Graduate work in Computer Science, Rutgers.
1979 Ph.D. in Computer Science, Rutgers.

Permanent Address:

66 Washington Place
Totowa Borough, New Jersey, 07512