The University of Alberta

IDENTITY AND SKOLEM FUNCTIONS IN RESOLUTION-BASED HYPOTHETICAL REASONING

by

George M. Ferguson

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Spring, 1989

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **Identity and Skolem Functions in Resolution-Based Hypothetical Reasoning** submitted by **George M. Ferguson** in partial fulfillment of the requirements for the degree of **Master of Science.**

.....

Supervisor

.....

.....

.....

Date

ABSTRACT

Conversion of first-order formulae to the clause form required for resolution theorem proving introduces Skolem terms intended to represent the individuals said to exist. These terms can lead to incorrect theories when assumptions are made concerning them. This has been called the reverse Skolemization problem (RSP) and is due to the lack of a precise semantics for the Skolem terms. We define *minimal attribution* as the desired interpretation of Skolem terms and show how, under this interpretation, the Skolem terms become *default names* for individuals described by their properties. We describe a set of intuitively satisfying solutions to the RSP which rule out incorrect theories and also show how to enforce minimal attribution in purely deductive systems. Both dynamic and static applications of the solution are considered, and a detailed description of an implementation of the former is provided.

Acknowledgements

I wish to acknowledge the invaluable help of my supervisor, Randy Goebel, who made me understand what was being said, especially since I was usually busy saying something else. Thanks also to the Theorist group at the University of Alberta, especially Scott Goodwin who tolerated me and provided valuable comments on this dissertation. Finally, thanks to the Power Plant crowd, especially Rod and Bill, for all the good times.

All the credit and none of the blame to my parents for getting me this far. Thanks.

> *The existence of different names for the same content is not always merely an irrelevant question of form; rather, that there are such names is the very heart of the matter if each is associated with a different way of determining the content.*
>
> Gottlob Frege, *Begriffschrifft* §8.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Existentially quantified variables have always posed problems for resolution-based deductive systems. In the process of converting an arbitrary formula to the required clause form, these variables are Skolemized to eliminate the existential quantifiers. While it can be shown that this transformation does not affect the unsatisfiability of the formula (the Skolem-Herbrand-Gödel Theorem), the resulting clauses may have interpretations which lead to incorrect conclusions. For example, naive interpretation of statements which are intended to mean that most individuals possess a property can conflict with information known explicitly about other objects when this information is expressed using existential quantifiers. Further, universally quantified variables in the antecedents of rules become Skolem terms during the conversion process and naive application of default rules to such terms can lead to incorrect assumptions.

We investigate the role of Skolem functions within the Theorist hypothetical reasoning framework which is based on a resolution theorem prover. Under an interpretation of *minimal attribution*, existential statements become, effectively, default rules for naming individuals which possess certain properties. We define minimal attribution and examine the problems involved in making consistent assumptions concerning Skolem individuals.

For purposes of illustration, we describe a progression of solutions to the reverse Skolemization problem. Each solution is intuitively reasonable, but only the last is capable of enforcing minimally attributive semantics in all situations involving assumptions. We characterize the difference between the solutions, and show how the final approach can be applied even in purely deductive situations when no assumptions are being made. We describe previous attempts to solve the RSP, and argue for the utility of our approach in assumption-based reasoning and for the validity of minimally attributive semantics for Skolem terms in general.

This dissertation is organized as follows. Chapter Two will set out the requisite logical preliminaries concerning clausal logic and will describe the Theorist hypothetical reasoning system. Chapter Three will outline the reverse Skolemization problem, presenting several examples of its occurance, and Chapter Four will present our solutions to the problem and several further examples. In Chapter Five we will cover the details of the implementation of the modified Theorist system and Chapter Six will summarize the results and outline potential future avenues of research.

Chapter 2

Logical Preliminaries

In this chapter we will set out the definitions and details of clausal logic and describe the Theorist hypothetical reasoning system. Further details concerning the former are available in any standard text on first-order logic, for example [Lov78], [GeN87], or [ChL73]. The definitive Theorist reference is [PGA87] but we will go into some detail here as the details are perhaps less well-known.

2.1. Clause form

Definition

A *term* is an *n*-ary function symbol followed by *n* arguments, each of which is either a term or a *variable*, where variables are not considered terms. In the case $n = 0$, the term is called a *constant*. We will denote variables by names starting with uppercase letters.

Definition

An *atomic sentence* is an *n*-ary predicate symbol followed by *n* arguments, each of which is a term or a variable. We will denote predicate, function and constant symbols by names starting with a lowercase letter, and trust that no confusion will result.

Definition

A *literal* is either an atomic sentence (*e.g.* `p(X)`) or the negation of an atomic sentence (written, *e.g.*, $\neg$`p(X)`). In the former case it is a *positive* literal, in the latter a *negative* one.

Definition

A *clause* is a disjunction of literals (written $l_1 \vee l_2 \vee \cdots \vee l_n$).

2.2. Conversion to clause form

Any well-formed formula (*wff*) of the first-order predicate calculus can be translated into a set of clauses using the following procedure.

1. Eliminate implication:

   $p \rightarrow q \equiv \neg p \vee q$.

2. Reduce scope of negation:

   $\neg \neg p \equiv p$,
   $\neg \forall X\, p(X) \equiv \exists X \neg p(X)$,
   $\neg \exists X\, p(X) \equiv \forall X \neg p(X)$,
   $\neg (p \wedge q) \equiv \neg p \vee \neg q$,
   $\neg (p \vee q) \equiv \neg p \wedge \neg q$.

3. Standardize variables apart:

   $\forall X\, (p(X) \wedge \exists X\, q(X)) \equiv \forall X\, (p(X) \wedge \exists Y\, q(Y))$.

4. Skolemize existentially quantified variables:

   $\forall X \exists Y\, p(Y) \equiv \forall X\, p(f(X))$, *(see below).*

5. Convert to prenex form:

   $\forall X\, p(X) \rightarrow \forall Y\, q(Y) \equiv \forall X \forall Y\, (p(X) \rightarrow q(Y))$.

6. Convert to conjunctive normal form:

   $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.

7. Drop universal quantifiers:

   $\forall X\, p(X) \equiv p(X)$.

8. Break conjuncts into clauses:

   $c_1 \wedge c_2 \wedge \ldots \wedge c_n \equiv \{c_1, c_2, \ldots, c_n\}$.

9. Rename variables:

   $\{p(X), q(X)\} \equiv \{p(X), q(Y)\}$.

For further details and examples, see [Nil80, §4.2.1]. Note that all clauses are implicitly universally quantified (step 7). We will generally ignore step 9; variables in different clauses should be assumed distinct even if their names are the same. We will also sometimes write clauses using implication signs to make their intended interpretation clearer.

The only step which does not involve only the use of straightforward logical identities or rewriting rules is step 4, in which we replace existentially quantified variables by functions which ‘‘represent’’ the individuals stated to exist. For example, the formula

$$\exists X \; \texttt{red(X)}$$

becomes

$$\texttt{red(c)}$$

where `c` is a *0*-ary Skolem function (a Skolem constant) denoting the individual said to be red. In the case that the existential quantifier occurs within the scope of universal quantifiers, we allow for the possibility that the individual referred to might depend on the values of the universally quantified variables. For example, from

$$\forall X \exists Y \; \texttt{is\_mother\_of(Y,X)}$$

we get

$$\forall X \; \texttt{is\_mother\_of(f(X),X)}$$

where `f(X)` is a Skolem term whose intended interpretation is that it denotes the mother of the individual `X`.

Note that symbols used to represent the Skolem individuals introduced by this step of the conversion must be distinct from any other constant or function symbols in the set of formulae being transformed; otherwise the resulting clause might mistakenly refer to an individual which other clauses already referred to.

2.3. Resolution theorem proving

The result that makes the clausal form of sentences attractive for theorem proving is the Skolem-Herbrand-Gödel Theorem which states, effectively, that if a set of formulae *S* is unsatisfiable, then the set of clauses created from *S* by the procedure described above is also unsatisfiable (*cf.* [Gal86, Chapters 7 and 8]).

In particular, the resolution proof procedure [Rob65] exploits this fact to construct proofs from sets of clauses. We will not go into the details of resolution refutations or proofs; see any of the references cited above for details. We will note however the advantages of resolution, since

the problems to be described later arise as a direct result of the use of clauses.

- Resolution is a correct and complete inference rule, therefore using it alone avoids the need to choose between alternative rules at each step of a proof. Systems that use multiple inference rules (*e.g.* natural deduction [BMN80] or Gentzen systems [Gal86]) often require heuristic or metareasoning procedures to decide what to do next.
- Implementation of resolution systems can be made efficient through careful use of pattern matching and database indexing techniques, see [Cam84]. The Japanese fifth-generation project (*cf.* [ICO88], [Fuc88]) has implemented resolution proving at the microcode level on special-purpose architectures.
- Horn clauses (clauses with at most a single positive literal) can be interpreted as procedure calls allowing natural and useful programs to be written using only clauses and resolution (*cf.* [Kow74], [vaK76]).
- Several tools are readily available for the development of clause-oriented systems, mostly derived from the PROLOG language [Col83] and its relatives. These provide robust platforms for prototyping resolution-based deductive systems, as opposed to the relative scarcity of other theorem provers. Further, PROLOG is a full-featured programming language implementing non-logical features through meta-predicates and procedural attachment of ‘‘system’’ predicates.

2.4. Theorist: A hypothetical reasoning system

Theorist [PGA87, vaG86] is a nonmonotonic reasoning system based on the scientific method and its ‘‘observe-hypothesize-predict-test’’ paradigm. Its knowledge base is partitioned into a set **F** of *facts*, which are taken to be true, and a set **H** of *hypotheses* (actually *hypothesis schemas*) instances of which may be used so long as there is no information to the contrary.

A Theorist database is said to *explain* a set of observations **O** if

$$\mathbf{F} \cup \mathbf{D} \models \mathbf{O},$$
$$\mathbf{F} \cup \mathbf{D} \text{ consistent}$$

for **D** a set of instances of **H**. In this case we say that **D** is the *theory* which explains **O**. 1.

1. The symbol **D** comes from ‘‘default theory’’, even though such usage is now somewhat misleading.

In general, Theorist is as powerful as any other default reasoning scheme, such as circumscription [McC80] or default logic [Rei80]. See [LiG89] for a comparison between Theorist and the former, and [Poo88a, Poo89] for comparisons with the latter.

2.5. Theorist example

The following Theorist database describes some possible commonsense knowledge about birds:

Example 2.1

$$\mathbf{H} = \{\, \texttt{birdsfly(X)}: \forall\, \texttt{X}\ \texttt{bird(X)} \rightarrow \texttt{flies(X)},$$
$$\texttt{emusdontfly(X)}: \forall\, \texttt{X}\ \texttt{emu(X)} \rightarrow \neg\, \texttt{flies(X)} \,\},$$
$$\mathbf{F} = \{\, \texttt{bird(tweety)},$$
$$\texttt{emu(edna)},$$
$$\forall\, \texttt{X}\ \texttt{emu(X)} \rightarrow \texttt{bird(X)} \,\}.$$

We name the hypotheses for ease of reference. 2. The intended interpretation of these statements should be clear: most birds fly; most emus don't fly; tweety is a bird; edna is an emu; all emus are birds. The precise meaning of ''most'' in this setting is an open question and is not relevant here.

We can explain the observation

$$\texttt{flies(tweety)}$$

with the theory

$$\{\, \texttt{birdsfly(tweety)} \,\},$$

meaning we use the default rule

$$\texttt{bird(X)} \rightarrow \texttt{flies(X)}$$

with reference to `tweety`. Note that we could not have derived `flies(tweety)` from the facts alone. We were able to use the default rule since there was no information to the contrary.

---

2. In fact, the hypothesis *name : formula* can be transformed into hypothesis *name* and fact *formula* $\leftarrow$ *name*. So long as *name* contains all universally quantified variables appearing in *formula* (which may be instantiated during a proof) the semantics are identical. See [Poo88a, §5] for further details.

2.6. Theorist and nonmontonicity

Clearly, we may be able to explain a given set of observations with several theories or explain both an observation and it's negation (with different theories, each of which is consistent despite being mutually inconsistent). Thus in the above example, we can explain

$$\texttt{flies(edna)}$$

with theory

$$\{\texttt{birdsfly(edna)}\}$$

and we can explain

$$\neg\texttt{flies(edna)}$$

with theory

$$\{\texttt{emusdontfly(edna)}\}.$$

This is an example of the multiple extension problem [HaM86] and is solved in the Theorist framework by *theory preference*: meta-rules which prune and order theories. So, for example, we might prefer the explanation of $\neg\texttt{flies(edna)}$ by virtue of a specificity criterion as in inheritance reasoning. For details, see [Poo85], [Kir87], or the other references listed at the end of this section.

In addition to straightforward explanation, we can use Theorist to *predict* or *conditionally explain*. For prediction, we predict those things which are logical consequences of preferred (for example, maximal) theories. A simple example is given by the ''Nixon diamond'' [THT87] which is axiomatized for Theorist as:

$$\mathbf{H} = \{\neg\texttt{pacifist(X)} \leftarrow \texttt{republican(X)},$$
$$\texttt{pacifist(X)} \leftarrow \texttt{quaker(X)}\},$$
$$\mathbf{F} = \{\texttt{republican(nixon)},$$
$$\texttt{quaker(nixon)}\}.$$

In other words, most republicans are not pacifists, most quakers are, and Nixon is both a quaker and a republican. If we predict only those things which are present in *all* extensions of a theory (and the facts) then we predict neither $\texttt{pacifist(nixon)}$ nor $\neg\texttt{pacifist(nixon)}$. If we predict those things present in *any* extension, then we predict both possiblities.

We conditionally explain observations **O** with conditions **C** if adding **C** to **F** would allow us to explain **O**. Conditional explanation is useful when there are no ‘‘traditional’’ explanations, however, if unrestricted, the exponential explosion of conditional explanations may be counter-productive, since any point at which a proof fails is a candidate for a condition. As before, theory preference information could be used to specify more ‘‘desirable’’ conditions. As a simple example, Theorist might be able to ask the value of something it cannot deduce, such as the result of a lab test in a diagnostic environment. Clearly these are good candidates for conditions, subject to cost and utility considerations.

The specification of Theorist makes no commitment as to how validity ($\models$) is to be implemented in order to actually build a system based on it’s principles. A reasonable approach is to implement validity by derivability ($\vdash$) using a complete proof procedure and use negation as failure [Cla78], [Llo87] to implement consistency checking. Thus from facts **F** and hypotheses **H** we explain **O** with theory **D** if

$$\mathbf{F} \cup \mathbf{D} \vdash \mathbf{O},$$
$$\mathbf{F} \cup \mathbf{D} \nvdash \neg d, \text{ for each } d \in \mathbf{D}.$$

To be even more specific, we implement derivability using a clausal theorem prover based on the MESON proof procedure [Lov78]. This is a resolution-based proof procedure which can be viewed as PROLOG with true negation and thus allows the use of arbitrary (as opposed to only Horn) clauses [UmP85], [Bow82]. Straightforward extensions allow the processing of disjunctive queries (*e.g.*, [Sti88]). We should point out that while validity is at best semi-decidable for first order logic (and thus for Theorist) the depth-first strategy of our implementation of the MESON procedure may cause it to pursue an infinite proof branch and miss solutions. In any event, these control issues are not of concern here since we are concerned with producing a correct specification (however see Chapter 5 for some discussion of this).

To conclude, among other areas of interest, Theorist can be used in planning and temporal reasoning to deal with the frame problem ([GoG87], [Goo87]), for analogical reasoning ([Jac86], [Goe89]), for diagnosis and abductive reasoning ([Poo86b], [Poo88b]), and in expert systems

([JoP85], [Tub86]).

Chapter 3

The Reverse Skolemization Problem

The reverse Skolemization problem (RSP) arises in any reasoning system based on clausal logic. In particular, it arises in Theorist when assumptions are made concerning Skolem individuals. In this chapter we describe the RSP from the point of view of theory formation in hypothetical reasoning and present examples of its occurance, allowing us to proceed to the solution described in Chapter Four. We will show how this is part of a larger problem concerning the identity of individuals said to exist, and will describe other approaches to the problem and their relative merits.

3.1. Example

Consider the following Theorist database, where the variables are intended to range over blocks in a ‘‘Blocks World’’ and the constants refer to particular blocks.

Example 3.1

$$\mathbf{H} = \{\, \forall X\ \mathtt{ontable(X)}\,\}, \tag{1}$$

$$\mathbf{F} = \{\, \mathtt{g} \leftarrow \exists Y\ (\mathtt{red(Y)} \wedge \mathtt{ontable(Y)}), \tag{2}$$

$$\exists Z\ \mathtt{red(Z)}, \tag{3}$$

$$\mathtt{red(a)}, \tag{4}$$

$$\neg\, \mathtt{ontable(a)}\,\}. \tag{5}$$

The intended interpretation of these formulae is

(1) We can assume any block is on the table unless we are told otherwise.

(2) g (our goal) is true if there is at least one block which is both red and on the table.

(3) There is (at least) one red block.

(4),(5) a is a block which is red and is not on the table.

The Herbrand universe [Gal86, §9.5.2] of this set of formulae consists solely of a (g is a constant predicate symbol).

We convert the database to clause form, following the procedure described in Chapter Two,

to obtain

$$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\}, \tag{1′}$$

$$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(Y)} \wedge \texttt{ontable(Y)}, \tag{2′}$$

$$\texttt{red(c)}, \tag{3′}$$

$$\texttt{red(a)}, \tag{4′}$$

$$\neg\texttt{ontable(a)} \,\} \tag{5′}$$

where $\texttt{c}$ is a Skolem constant. The Herbrand universe of the set of clauses is $\{\, \texttt{a}, \texttt{c} \,\}$. In what follows, we will often want to refer to the set of individuals explicitly named in the non-clausal form of our theories, as distinct from those created by Skolemization.

Definition

The *non-Skolem Herbrand universe* (*NS-Herbrand universe* or *NSHU*) of a set of clauses **C** is the set of non-Skolem terms in the Herbrand universe of **C**.

Clearly, if a set of clauses **C** is derived from a set of formulae **F** by the process described in chapter two, the NS-Herbrand universe of **C** will be the same as the Herbrand universe of **F**.

Now consider explaining $\texttt{g}$. We can't explain it with $\texttt{Y} = \texttt{a}$ due to (5′). We have $\texttt{red(c)}$ however, from (3′), and could explain $\texttt{g}$ if we could assume $\texttt{ontable(c)}$ (since we can't establish it from **F**). The attempt to show $\neg\texttt{ontable(c)}$ fails, so it is consistent to make the assumption, thus explaining $\texttt{g}$ with theory $\{\, \texttt{ontable(c)} \,\}$. The proof is depicted graphically in Figure 3.1. The dashed arrow indicates a consistency check.

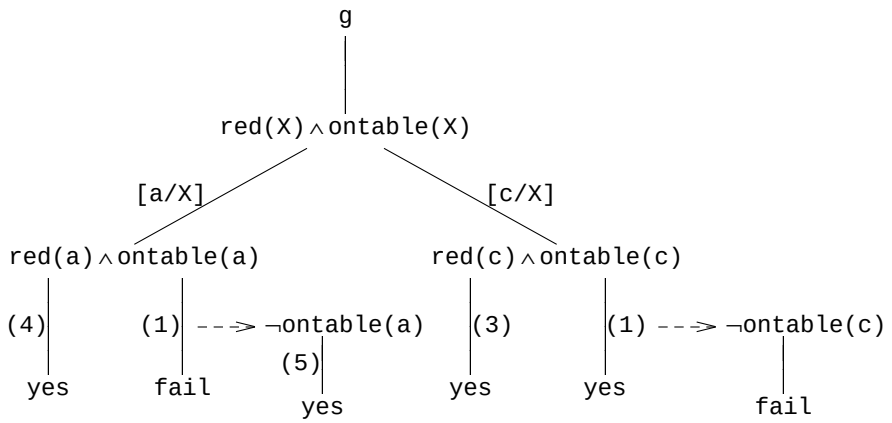

Figure 3.1 Theorist explanation of g

3.2. The meaning of existentials

In order to evaluate the answer of Example 3.1 with respect to the intended meaning of the original formulae, we must determine more precisely the semantics of existentially quantified statements in our system.

On the one hand, if we use such statements *referentially*, the fact (3) then refers to a specific individual which we are expected to be already familiar with. To properly add this fact to our database would require that we somehow identify the individual in order to attribute the new property (redness) to it. Under this interpretation, Skolem terms do not appear. If a statement fails to refer (*i.e.* we can't find its referent) then the statement is an error (either meaningless or false depending on your philosophical preferences).

Within the framework of a commonsense or default reasoning system such as Theorist however, it makes more sense to interpret existential statements *attributively*, that is without commitment to a particular individual. In this sense then, we are simply stating that (at least one) such object exists, without indicating whether we are familiar with it or not. Furthermore, the Skolemization procedure provides a name for the existing individual.

We will further want to impose a *minimality* condition on the use of posited individuals: an existential statement only commits to the existence of a single satisfying individual possessing the specified properties, while not, of course, ruling out others. We observe that this notion is also

embodied by the Skolemization procedure which replaces existentially quantified variables with functions which denote single individuals. 3. We call this intuitively satisfying interpretation of existential statements *minimally attributive*. The rest of this dissertation is concerned with enforcing this notion of minimal attribution.

To illustrate minimal attribution, we first observe that Skolem terms can arise in two different ways in our clausal database. These are shown in Table 3.1 which shows the original, non-clausal statements and their Skolemized, clausal forms.

$\forall X\, p(X) \# p(X)$
$\exists X\, p(X) \# p(c_1)$
$g \leftarrow \forall X\, p(X) \# g \leftarrow p(c_2)$
$g \leftarrow \exists X\, p(X) \# g \leftarrow p(X)$

Table 3.1 Two uses of Skolem terms

The first type of use of Skolem terms is *assertional*, typified by the second line of Table 3.1. In this case, we are asserting that an individual with certain properties exists, and, under a minimally attributive interpretation, the Skolemized form of the statement becomes a default rule for naming such an individual. In order to preserve the desired semantics, such a rule (statement) should not be used when existing individuals in the NS-Herbrand universe of our database satisfy the definition.

The second use of Skolem terms is *verificational*, typified by the third line of Table 3.1. Here the Skolem term is a ‘‘placeholder’’ whose uniqueness (see page 5) ensures that only a universally quantified assertion will prove it. This is correct, as the original statement had a universally quantified formula as antecedent. In keeping with the minimally attributive semantics, Skolem terms used verificationally should not appear in the results of queries, and when they appear in theories they should be read as universally quantified variables.

3. Thus the correct way to say ‘‘Everyone has at least one parent’’ is not $\forall X \exists Y\, \mathtt{parent(x,y)}$, at least not if two possible parents are desired. Using this formula, it is possible that X has two parents, *i.e.*, it is not ruled out, however the function introduced to replace Y refers to one or the other parent but not to both. While ‘‘at least one’’ would seem to subsume ‘‘two’’, the Skolemization procedure does not reflect this.

As in [Poo86a], we can record the (minimally attributive) definitions of Skolem constants using Hilbert's ε-operator [Lei69]. For an arbitrary formula

$$\forall X_1 \cdots \forall X_n \exists Y\, p(X_1, \ldots, X_n, Y)$$

which when Skolemized becomes

$$\forall X_1 \cdots \forall X_n\, p(X_1, \ldots, X_n, f(\bar{X})),$$

we record the definition of the Skolem term f as

$$f(\bar{X}) = \varepsilon Y\, .\, p(X_1, \ldots, X_n, Y).$$

We call the formula $p(X_1, \ldots, X_n, Y)$ the *defining sentence* of f. In certain formalisms, namely DLOG [Goe85], the ε-expression is a term and can appear in formulae itself. We will not use it as such however: for us it is simply an extra-linguistic expression used to record definitions. We will see in the next chapter how these definitions can be effectively used.

3.3. The reverse Skolemization problem

Under the minimally attributive interpretation of existentials then, the answer arrived at by Theorist in Section 3.1 is obviously wrong. The definition of c is

$$\varepsilon X\, .\, red(X).$$

Since a satisfies the definition of c, that is

$$\mathbf{F} \models red(a),$$

we should not use (3) to assume the existence of another individual c with those properties. Recall that c is an assertional Skolem term, hence our preferred, minimally attributive answer is that g cannot be explained.

In general, we should not use clauses containing Skolem individuals when elements of the NS-Herbrand universe of the clauses satisfy the definitions of the Skolem individuals. In this way we preserve the minimally attributive aspect of existential statements. If no member of the NS-Herbrand universe satisfies the definition of a Skolem function, then it is legitimate to consider the new (Skolem) individual.

To further illustrate the point, suppose we replace (4′) in Example 3.1 to get

Example 3.2

$$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\}, \tag{1′}$$

$$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{red(Y)} \wedge \mathtt{ontable(Y)}, \tag{2′}$$

$$\mathtt{red(c)}, \quad \mathtt{c} = \varepsilon \mathtt{X} \,.\, \mathtt{red(X)}, \tag{3′}$$

$$\neg \mathtt{red(a)}, \tag{4′}$$

$$\neg \mathtt{ontable(a)} \,\}. \tag{5′}$$

That is, the block $\mathtt{a}$ is now known to be both non-red and not on the table. The explanation of $\mathtt{g}$ proceeds exactly as before, yielding theory

$$\{\, \mathtt{ontable(c)} \,\}.$$

This time however, there is no element of the NS-Herbrand universe of $\mathbf{F}'$ which satisfies the definition of $\mathtt{c}$. As a result, we must use (3′) to posit a new individual, namely $\mathtt{c}$, which can consistently be assumed to be on the table, and the explanation of $\mathtt{g}$ is correct.

As a final example, consider the following database which illustrates the verificational use of Skolem terms:

Example 3.3

$$\mathbf{H} = \{\, \forall \mathtt{X}\, \mathtt{ontable(X)} \,\}, \tag{1}$$

$$\mathbf{F} = \{\, \mathtt{g} \leftarrow \forall \mathtt{X}\, \mathtt{ontable(X)}, \tag{2}$$

$$\neg \mathtt{ontable(a)} \,\}, \tag{3}$$

which, when converted to clauses, yields

$$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\}, \tag{1′}$$

$$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{ontable(c)}, \tag{2′}$$

$$\neg \mathtt{ontable(a)} \,\}, \tag{3′}$$

where $\mathtt{c}$ is a verificational Skolem constant whose definition is

$$\mathtt{c} = \varepsilon \mathtt{X} \,.\, \neg \mathtt{ontable(X)}.$$

Clearly we will be able to explain $\mathtt{g}$ with theory

$$\{\, \mathtt{ontable(c)} \,\}$$

despite the fact that not all known blocks are on the table (3′). Observe that if we had

$$\mathtt{ontable(a)} \tag{3′}$$

instead, this answer would be correct, since we know of no blocks which are not on the table, and

the hypothesis allows us to conclude that any others are on the table. In fact, if we didn't have 3′ at all, the answer would be trivially correct as there are no blocks. (*i.e.*, the NS-Herbrand universe is empty.)

The reason for Theorist's failure is equally clear: the clausal theorem prover implicitly contains the *unique names hypothesis*

$$\neg \mathtt{eq}(\alpha, \beta) \wedge \neg \mathtt{eq}(\beta, \gamma) \wedge \neg \mathtt{eq}(\alpha, \gamma) \cdots$$

extended for all distinct terms $\alpha, \beta, \gamma, \ldots$. Our equality predicate $\mathtt{eq}$ is used in the standard Fregean sense of *semantic identity*: two terms are equal if and only if they represent the same individual in the semantic domain. The unique names assumption means that Theorist implicitly assumes that different constants and functions represent different individuals in the domain. For those individuals posited to exist by the Skolemization process however, this may not be the case if minimally attributive semantics are to be preserved.

3.4. Other approaches

Before describing our solution to the RSP in the next chapter, we should briefly describe other approaches and solutions.

First, Cox and Pietrzkowski [CoP84] proposed an algorithm for reversing the Skolemization of a set of clauses which is based syntactic transformations that replace each clause in the set by a set of formulae. However, as the results are full first-order formulae and not clauses, this technique does not apply to our resolution-based approach, except perhaps as a way of making answers containing Skolem functions more palatable. Since the mapping of clauses to formulae is not (cannot be due to ambiguity in resolving quantifier scoping) one-to-one, even this is of questionable value within our framework. See also [Nil80] for details of answer-extraction involving Skolem functions.

Early work on the problem within the Theorist project was done by Poole [Poo86a] who proposed a proof-procedural solution. Basically, this relied on asserting new facts concerning individuals which satisfied the definitions of Skolem individuals and then re-explaining the goal with a different theory. While Poole's work formed the foundation of our solution and, we

believe, correctly identified the root of the RSP, it clouded the essential question of the semantics of Skolem individuals with respect to other constants and functions. Our work can be viewed as a clarification (and extension) of Poole's; all the examples from [Poo86a] are done in Section 5.6.

Finally, we should note that, of course, the problem wouldn't exist if we were using a non-clause-based theorem prover, such as a natural deduction system [BMN80, Gal86] to implement validity. However, as pointed out in Chapter Two, the benefits of resolution with respect to both speed and usability justify the use of clauses and the resulting attention that must be paid to problems such as the RSP.

Chapter 4

Identity and Skolem Functions

In the previous chapter we observed that Theorist can form incorrect theories when dealing with existential formulae, at least with respect to a given interpretation of these statements. We observed that the problem was a manifestation of a more general problem concerning the identity of Skolem terms arising either assertationally or verificationally. We also saw that the minimally attributive interpretation of existential formulae is intuitively appropriate in hypothetical reasoning situations.

In this chapter we describe our solution to the reverse Skolemization problem. In order to demonstrate the sufficiency and necessity of our approach, the exposition will proceed in stages: each section will solve a larger part of the problem. After showing how the final solution rules out all incorrect theories, we will describe how it can be applied even in purely deductive situations to enforce minimal attribution with respect to Skolem terms. We describe how this final solution can be easily incorporated into the existing Theorist framework.

The progression of solutions to the RSP can be viewed as a *dynamic* (or interpreted) enforcement of the minimally attributive semantics of existential statements, so we describe how a similar effect could be obtained by ''compiling'' the intended interpretation into a *static* database. This alternative is viewed as a traditional issue in nonmonotonic reasoning: any nonmonotonic reasoning system must decide on either a static or a dynamic approach to database management.

4.1. Negating Skolem terms

In Example 3.3, we observed that Theorist's failure to get the correct answer was due to its use of the statement

$$g \leftarrow \forall X \; \texttt{ontable(X)}$$

which introduced a verificational Skolem term when transformed to

$$g \leftarrow \texttt{ontable(c)}$$

to create a new individual `c` when our database **F** already contained an individual (`a`) satisfying

the definition of c. We noted that the problem arose due to Theorist's incorrect use of the hypothesis ontable(X). Now recall that to show consistency, Theorist attempts to show that the negation of the hypothesis under consideration follows from the facts, 4. in this case that

$$\mathbf{F} \models \neg \texttt{ontable(c)}.$$

If the attempt fails, as it does in this case, then the hypothesis can be consistently assumed.

Our first solution to the RSP amounts to a naive attempt at reversing the Skolemization of the hypothesis under consideration. We observe that

$$\neg \texttt{ontable(c)}$$

is not ''really'' the negation of

$$\texttt{ontable(c)}$$

when c is a Skolem constant. The desired interpretation of the latter statement is

$$\exists \texttt{X ontable(X)},$$

which when negated and converted to clauses (Section 2.2) would yield

$$\neg \texttt{ontable(X)}.$$

Therefore, before attempting to show an inconsistency we must replace each Skolem term in the hypothesis we are considering with a unique variable (the same variable for each occurance of the term). 5.

To properly enforce minimally attributive semantics, if we succeed in showing this modified negation, we must then test that the individual instantiating this variable is in the NS-Herbrand universe of our database. Otherwise we might, for example, simply use the Skolem term we replaced in the first place. An effect of this is that a database which is redundant with respect to existential statements such as

$$\mathbf{F} = \{ \exists \texttt{X p(X)}, \\ \exists \texttt{Y p(Y)} \}$$

will not be interpreted properly: we should, from a minimal attribution standpoint, get one

4. Actually, from the facts and the current theory, but we can safely ignore any current theory for demonstration purposes.

5. Note that since the consistency check amounts to a query, the variable is existential, not universal, and will be instantiated by the resolution proof procedure.

answer, but the procedure described above will not find any. In this case, an arbitrary choice between the two Skolem terms resulting from converting such a database to clauses must be made. Having mentioned this problem we will ignore it, but see Section 4.6 for a discussion of related issues.

Reconsider Example 3.3:

$$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\}, \tag{1'}$$

$$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{ontable(c)},\ \ \mathtt{c} = \varepsilon \mathtt{X} \,.\, \neg \mathtt{ontable(X)}, \tag{2'}$$

$$\neg \mathtt{ontable(a)} \,\}. \tag{3'}$$

This time rather than trying to show that

$$\mathbf{F} \models \neg \mathtt{ontable(c)}$$

we will try to show that

$$\mathbf{F} \models \exists \mathtt{Z} \,(\neg \mathtt{ontable(X)} \wedge \mathtt{X} \in \mathit{NSHU}),$$

which is true, for $\mathtt{X} = \mathtt{a}$. Thus the consistency check fails, the hypothesis is rejected, and the goal $\mathtt{g}$ is not explained, correctly.

As another example, consider the use of hypotheses to specify the *closed-world assumption* [Rei81] as in [Poo86a, Example 3]:

Example 4.1

$$\mathbf{H} = \{\, \forall \mathtt{X} \forall \mathtt{Y} \neg \mathtt{on(X,Y)} \,\}, \tag{1}$$

$$\mathbf{F} = \{\, \forall \mathtt{X} \,(\mathtt{cleartop(X)} \leftarrow \neg \exists \mathtt{Y}\, \mathtt{on(Y,X)}), \tag{2}$$

$$\mathtt{on(a,b)} \,\}. \tag{3}$$

The hypothesis means that the only blocks that are on each other are those explicitly stated to be so in the facts; all others can be assumed not to be on anything and not to have anything on them. The facts define what it means to have a clear top, and describe that block $\mathtt{a}$ is on block $\mathtt{b}$. Converting to clauses yields

$$\mathbf{H}' = \{\, \neg \mathtt{on(X,Y)} \,\}, \tag{1'}$$

$$\mathbf{F}' = \{\, \mathtt{cleartop(X)} \leftarrow \neg \mathtt{on(f(X),X)},\ \ \mathtt{f(X)} = \varepsilon \mathtt{Y} \,.\, \mathtt{on(Y,X)}, \tag{2'}$$

$$\mathtt{on(a,b)} \,\}. \tag{3'}$$

As before, the RSP is manifested in the fact that we can explain both $\mathtt{cleartop(a)}$ and

cleartop(b) with theories { ¬ on(f(a),a) } and { ¬ on(f(b),b) } respectively, since neither of these can be proven inconsistent, despite our explicit knowledge that b does not have a clear top since a is on it.

Applying our solution however, in testing the consistency of

$$\neg \texttt{on(f(a),a)}$$

we try to show that

$$\mathbf{F} \models \exists \texttt{Z}\,(\texttt{on(Z,a)} \wedge \texttt{Z} \in \mathit{NSHU})$$

rather than

$$\mathbf{F} \models \texttt{on(f(a),a)}.$$

This fails, so we correctly conclude that a has a clear top. The consistency check for

$$\neg \texttt{on(f(b),b)}$$

becomes

$$\mathbf{F} \models \exists \texttt{Z}\,(\texttt{on(Z,b)} \wedge \texttt{Z} \in \mathit{NSHU})$$

which succeeds, for Z = a, thus we correctly cannot explain b having a clear top.

We comment that if the syntactic manipulations involved in this solution seem disquieting, the same effect can be acheived if the implementation of validity includes equality (as before, in the Fregean sense of semantic identity) as in [RoW69], [Yuk87], or [Kor83]. If so, then to acheive the solution described above we allow Theorist (during consistency checking) to assume that a Skolem term is equal to any term in the NS-Herbrand universe. We then have a hierarchy of assumptions, where the ‘‘normal’’ assumptions can be ruled out by *ehypotheses*: assumptions of equality between a Skolem and a non-Skolem term, which can themselves only be ruled out by the facts.

To illustrate, for Example 3.3 we generate an ehypothesis for the Skolem term c to get

$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\},$

$\mathbf{H}'_{\mathbf{e}} = \{\, \texttt{eq(c,Z)} \leftarrow \texttt{Z} \in \mathit{NSHU} \,\},$

$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{ontable(c)},$

$\neg \texttt{ontable(a)} \,\}.$

When we try to explain g we have to test the consistency of the assumption ontable(c) by

trying to show that

$$\mathbf{F} \cup \mathbf{D_e} \models \neg \texttt{ontable(c)},$$
$$\mathbf{F} \cup \mathbf{D_e} \text{ consistent}$$

for $\mathbf{D_e}$ a set of instances of $\mathbf{H_e}$. In the example, using the substitutivity axiom for `ontable` in its contrapositive form

$$\neg \texttt{ontable(X)} \leftarrow \texttt{eq(X,Y)} \wedge \neg \texttt{ontable(Y)}$$

and the fact `ontable(a)`, we will succeed if it is consistent to assume the ehypothesis. Since

$$\mathbf{F} \not\models \neg \texttt{eq(c,a)}$$

we can make the equality assumption, therefore proving $\neg$`ontable(c)` and ruling out the explanation of `g`, correctly.

We will not pursue this avenue any further (however see Section 5.3) since the overhead of deduction with equality is too high when we can acheive our goals by performing meta-manipulations instead.

4.2. Restoring the context of Skolem terms

The preceding solution works so long as we are always making assumptions concerning the defining predicates of Skolem terms. That is, in Example 3.3 we are assuming

$$\texttt{ontable(c)}$$

for

$$\texttt{c} = \varepsilon \texttt{X} . \neg \texttt{ontable(X)}$$

and in Example 4.1 we are assuming

$$\neg \texttt{on(f(a),a)}$$

for

$$\texttt{f(X)} = \varepsilon \texttt{Y} . \texttt{on(Y,X)}.$$

Example 3.2 illustrates the inadequacy of this solution for general assumptions concerning Skolem terms. Recall that we had

$$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\}, \tag{1'}$$

$$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(Y)} \wedge \texttt{ontable(Y)}, \tag{2'}$$

$$\texttt{red(c)}, \quad \texttt{c} = \varepsilon \texttt{X} \,.\, \texttt{red(X)}, \tag{3'}$$

$$\neg\, \texttt{red(a)}, \tag{4'}$$

$$\neg\, \texttt{ontable(a)} \,\}. \tag{5'}$$

and that we could explain $\texttt{g}$ with theory $\{\, \texttt{ontable(c)} \,\}$ since

$$\mathbf{F} \not\models \neg\, \texttt{ontable(c)}.$$

This was correct since $\texttt{a}$ did not satisfy the definition of $\texttt{c}$.

Applying the previous solution, however, requires that we test whether

$$\mathbf{F} \models \exists \texttt{Z} \,(\neg\, \texttt{ontable(Z)} \wedge \texttt{Z} \in \mathit{NSHU})$$

instead. Clearly this is the case, for $\texttt{Z} = \texttt{a}$, so the hypothesis will be rejected, incorrectly. The problem is that we have ignored the definition of the Skolem term and allowed the assumption concerning it to be ruled out by a fact concerning an individual which does not satisfy the definition.

Our second solution corrects this problem by requiring that, in order to rule out a hypothesis concerning a Skolem term, not only do we have to show its modified negation as in the previous section, but we must also show that the individual instantiating the new variable satisfies the definition of the Skolem term.

Thus in the previous example, rather than showing that

$$\mathbf{F} \models \exists \texttt{Z} \,(\neg\, \texttt{ontable(Z)} \wedge \texttt{Z} \in \mathit{NSHU})$$

we try to show that

$$\mathbf{F} \models \exists \texttt{Z} \,(\neg\, \texttt{ontable(Z)} \wedge \texttt{red(Z)} \wedge \texttt{Z} \in \mathit{NSHU})$$

since

$$\texttt{c} = \varepsilon \texttt{X} \,.\, \texttt{red(X)}.$$

We cannot show this, so the hypothesis is accepted and the query $\texttt{g}$ explained, correctly, with theory $\{\, \texttt{ontable(c)} \,\}$.

As another example, reconsider Example 3.1:

$$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\}, \tag{1'}$$

$$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(Y)} \wedge \texttt{ontable(Y)}, \tag{2'}$$

$$\texttt{red(c)}, \tag{3'}$$

$$\texttt{red(a)}, \tag{4'}$$

$$\neg\texttt{ontable(a)} \,\} \tag{5'}$$

This time we will show the hypothesis $\texttt{ontable(c)}$ inconsistent since

$$\mathbf{F} \models \neg\texttt{ontable(a)} \wedge \texttt{red(a)} \wedge \texttt{a} \in \mathit{NSHU},$$

thus correctly not allowing us to explain $\texttt{g}$.

Note that in the two examples in Section 4.1, the negation of the hypothesis being considered and the definition of the Skolem term concerned were identical. These can therefore be considered a special case of this section's solution and they would clearly both be handled properly.

In the alternative approach based on ehypotheses, this solution requires that we only assume Skolem terms equal to terms in the NS-Herbrand universe which satisfy their definitions. Thus our ehypotheses would be of the form

$$\texttt{eq(c,X)} \leftarrow \textit{defn. of}\ \texttt{c[X/c]} \wedge \texttt{X} \in \mathit{NSHU},$$

where the antecedent should be read as ''the definition of $\texttt{c}$ with $\texttt{X}$ substituted for $\texttt{c}$'' where $\texttt{X}$ is a unique variable. In the example from the previous section our ehypothesis becomes

$$\mathbf{H}'_{\mathbf{e}} = \{\, \texttt{eq(c,Z)} \leftarrow \texttt{red(Z)} \wedge \texttt{Z} \in \mathit{NSHU} \,\}.$$

As in that section, we will not pursue this approach any further.

4.3. Ensuring minimally attributive theories

The previous solution seems to capture the intuition regarding minimally attributive Skolem terms. In this section however, we will see that it does not always rule out theories which unnecessarily introduce Skolem terms.

For example, consider the following weakening of the ''blocks world'' database from Section 3.1:

Example 4.2

$$\mathbf{H} = \{\, \forall X \; \texttt{ontable(X)} \,\},$$

$$\mathbf{F} = \{\, \texttt{red(a)},$$

$$\exists X \; \texttt{red(X)} \,\}.$$

Recall that in the original example we also had the fact

$$\neg \texttt{ontable(a)}$$

but here we do not know whether $\texttt{a}$ is on the table or not. We convert to clauses to get

$$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\},$$

$$\mathbf{F}' = \{\, \texttt{red(a)},$$

$$\texttt{red(c)}, \;\; \texttt{c} = \varepsilon X \,.\, \texttt{red(X)} \,\}.$$

Now consider explaining the same query as before:

$$? \leftarrow \texttt{red(X)} \wedge \texttt{ontable(X)}.$$

We clearly get the (correct) answer $\texttt{X} = \texttt{a}$ with theory

$$\{\, \texttt{ontable(a)} \,\}.$$

Unfortunately we also get the answer $\texttt{X} = \texttt{c}$ with theory

$$\{\, \texttt{ontable(c)} \,\}$$

since there is no way of deriving $\neg \texttt{ontable(X)}$ from **F**. Theorist produces an answer which postulates the existence of a Skolem individual despite the fact that there are individuals in its database which satisfy the individual's definition, thus violating our criterion of minimal attribution. Note that we get the same effect even if we know that $\texttt{a}$ is on the table since, again, we have no negative information which would allow us to rule out a hypothesis.

From this we conclude that our modified consistency checking criterion, which requires us to show both the modified negation of the hypothesis under consideration *and* the definition of the Skolem term involved, is too strong. In order to preserve minimally attributive semantics, it is only necessary to show, in order to rule out a hypothesis concerning a Skolem term, that its definition is satisfied by another individual. Of course, we must still test the consistency of the assumption *normally* (by attempting to show its negation) in case we have contradictory explicit information about the Skolem term.

So in the previous example (4.2), when we are considering assuming

$$\texttt{ontable(c)}$$

we test whether

$$\mathbf{F} \models \exists \texttt{Z}\,(\texttt{red(X)} \wedge \texttt{X} \in \textit{NSHU}) \vee \neg\,\texttt{ontable(c)}.$$

The first disjunct is the definition of the Skolem term $\texttt{c}$, the second is the simple consistency check for the assumption. Clearly

$$\mathbf{F} \models \texttt{red(a)} \wedge \texttt{a} \in \textit{NSHU}$$

thus correctly ruling out the hypothesis. Theorist produces only the (correct) theory $\{\,\texttt{ontable(a)}\,\}$. As in Example 3.2, if we know instead that $\texttt{a}$ was non-red, the test would fail, the hypothesis would be accepted, and we would get the correct explanation with theory $\{\,\texttt{ontable(c)}\,\}$.

An example of how the simple consistency check is still necessary is provided by the following:

Example 4.3

$$\mathbf{H} = \{\,\forall \texttt{X}\ \texttt{ontable(X)}\,\},$$

$$\mathbf{F} = \{\,\texttt{g} \leftarrow \exists \texttt{X}\,(\texttt{red(X)} \wedge \texttt{ontable(X)}),$$

$$\exists \texttt{X}\,(\texttt{red(X)} \wedge \neg\,\texttt{ontable(X)})\,\}$$

which, when converted to clauses, yields

$$\mathbf{H}' = \{\,\texttt{ontable(X)}\,\},$$

$$\mathbf{F}' = \{\,\texttt{g} \leftarrow \texttt{red(X)} \wedge \texttt{ontable(X)},$$

$$\texttt{red(c)},\ \ \texttt{c} = \varepsilon \texttt{X}\,.\,\texttt{red(X)} \wedge \neg\,\texttt{ontable(X)},$$

$$\neg\,\texttt{ontable(c)},\ \ \texttt{c} = \varepsilon \texttt{X}\,.\,\texttt{red(X)} \wedge \neg\,\texttt{ontable(X)}\,\}.$$

Clearly then, without the simple consistency check we would explain $\texttt{g}$ with theory $\{\,\texttt{ontable(c)}\,\}$ despite the fact that we are told only about a block which is explicity *not* on the table.

We conclude this section by presenting the examples of the previous two sections and showing how they are all solved using this last solution. Further examples, including all of those in [Poo86a], are presented in Section 4.5.

Example 3.3

$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\},$

$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{ontable(c)},\ \ \texttt{c} = \varepsilon \texttt{X} \,.\, \neg \texttt{ontable(X)},$

$\neg \texttt{ontable(a)} \,\}.$

In trying to explain $\texttt{g}$, we try to assume $\texttt{ontable(c)}$. To rule out this assumption we must show that

$$\mathbf{F} \models \exists \texttt{Z}\,(\neg \texttt{ontable(X)} \wedge \texttt{X} \in \mathit{NSHU}) \vee \neg \texttt{ontable(c)}.$$

We show the condition, for $\texttt{X} = \texttt{a}$, thus correctly ruling out the hypothesis. There is no other explanation of $\texttt{g}$.

Example 4.1

$\mathbf{H}' = \{\, \neg \texttt{on(X, Y)} \,\},$

$\mathbf{F}' = \{\, \texttt{cleartop(X)} \leftarrow \neg \texttt{on(f(X), X)},\ \ \texttt{f(X)} = \varepsilon \texttt{Y} \,.\, \texttt{on(Y, X)},$

$\texttt{on(a, b)} \,\}.$

Explaining $\texttt{cleartop(a)}$ requires assuming $\neg \texttt{on(f(a), a)}$. We test whether

$$\mathbf{F} \models \exists \texttt{Y}\,(\texttt{on(Y, a)} \wedge \texttt{Y} \in \mathit{NSHU}) \vee \texttt{on(f(a), a)}$$

We fail to show either disjunct, thus accepting the hypothesis and explaining $\texttt{cleartop(a)}$ with theory $\{\, \neg \texttt{on(f(a), a)} \,\}$.

To explain $\texttt{cleartop(b)}$ we try to assume $\neg \texttt{on(f(b), b)}$ which requires testing whether

$$\mathbf{F} \models \exists \texttt{Y}\,(\texttt{on(Y, b)} \wedge \texttt{Y} \in \mathit{NSHU}) \vee \texttt{on(f(b), b)}.$$

This is true, for $\texttt{Y} = \texttt{a}$, so the hypothesis is ruled out and we cannot explain $\texttt{ontable(b)}$, correctly.

Example 3.2

$\mathbf{H}' = \{\, \texttt{ontable(X)} \,\},$

$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(Y)} \wedge \texttt{ontable(Y)},$

$\texttt{red(c)},\ \ \texttt{c} = \varepsilon \texttt{X} \,.\, \texttt{red(X)},$

$\neg \texttt{red(a)},$

$\neg \texttt{ontable(a)} \,\}.$

We can explain g if we can show that the assumption ontable(c) is consistent, which requires that we test whether

$$\mathbf{F} \models \exists X (\texttt{red}(X) \wedge X \in \mathit{NSHU}) \vee \neg \texttt{ontable(c)}.$$

We cannot show this, so g can be explained with theory { ontable(c) }.

Example 3.1

$\mathbf{H}' = \{$ ontable(X) $\},$

$\mathbf{F}' = \{$ g $\leftarrow$ red(Y) $\wedge$ ontable(Y),

red(c),

red(a),

$\neg$ ontable(a) $\}$

Again we want to use the assumption ontable(c) to explain g, so we test whether

$$\mathbf{F} \models \exists X (\texttt{red}(X) \wedge X \in \mathit{NSHU}) \vee \neg \texttt{ontable(c)}.$$

This is true, for X = a, so we reject the hypothesis correctly and cannot explain g.

4.4. Minimal attribution in general

We have shown how the solution to the RSP presented in the previous section eliminates all incorrect theories involving both assertational and verificational Skolem terms, and ensures minimally attributive theories. We may however want to take this further and enforce minimally attributive semantics even in purely deductive situations. Our previous solution can be applied in these circumstances almost directly.

Consider the following deductive Theorist database (no hypotheses), which states that everyone has a mother and that wilma is the mother of pebbles:

Example 4.4

$\mathbf{F} = \{ \forall X \exists Y$ mother–of(X, Y),

mother–of(pebbles, wilma) $\}$.

Conversion to clauses yields

$$\mathbf{F}' = \{\texttt{mother\_of(X,m(X))},\ \texttt{m(X)} = \varepsilon \texttt{Y}\,.\,\texttt{mother\_of(X,Y)},$$
$$\texttt{mother\_of(pebbles,wilma)}\}.$$

Now if we query Theorist about `pebbles`' mother, that is

$$?\leftarrow \texttt{mother\_of(pebbles,Z)},$$

we get the *two* deductive solutions

$$\texttt{Z} = \texttt{wilma}$$

and

$$\texttt{Z} = \texttt{m(pebbles)}.$$

This second answer violates our desired minimally attributive semantics since we already know of an individual (`wilma`) which satisfies the definition of `m(pebbles)`.

To adapt our solution to deductive situations such as this requires converting the existential statement

$$\forall \texttt{X} \exists \texttt{Y}\ \texttt{mother\_of(X,Y)}$$

to the meta-level conditional

$$\texttt{mother\_of(X,m(X))} \leftarrow \mathbf{F} \nvDash\ \exists \texttt{Z}\,(\texttt{mother\_of(X,Z)} \wedge \texttt{Z} \in \textit{NSHU}).$$

This will clearly succeed in ruling out the unwanted second solution. In addition, we will also be able to correctly answer the query

$$?\leftarrow \texttt{mother\_of(wilma,Z)}$$

with

$$\texttt{X} = \texttt{m(wilma)}$$

since we know of no other individual which is `wilma`'s mother, that is

$$\mathbf{F} \nvDash\ \exists \texttt{Z}\,(\texttt{mother\_of(wilma,Z)} \wedge \texttt{Z} \in \textit{NSHU}).$$

This solution also make intuitive sense: we are explicitly saying that Skolem terms can only be introduced in a minimally attributive manner, that is, when their definitions are not satisfied by non-Skolem individuals. We also should note here that the general solution is certainly not difficult to implement. As we will describe in Chapter Five, we have already implemented $\models$ in order to build theories and test consistency. As we do there, we can implement $\nvDash$ by failing to show $\models$.

Clearly this is closely related to our solution from the previous section for explanations requiring assumptions. If we translate all statements involving Skolem terms to the meta-level form described above and return to simple consistency checking as in the original Theorist, we will completely incorporate this solution. To illustrate, observe that the statement

$$\exists X\, p(X)$$

becomes

$$p(c) \leftarrow \mathbf{F} \not\models \exists X (p(X) \wedge X \in \mathit{NSHU})$$

according to the specification of our solution, and when trying to assume

$$q(c)$$

we have to show simply that

$$\mathbf{F} \not\models \neg q(c)$$

regardless of the fact that c is a Skolem term. Thus, as for the solution in Section 4.3, showing either

$$\mathbf{F} \models \exists X (p(X) \wedge X \in \mathit{NSHU})$$

or

$$\mathbf{F} \models \neg q(c)$$

will rule out the hypothesis $q(c)$; in the first case the antecedent of the modified fact is not satisfied and in the second the hypothesis is simply inconsistent.

We will again conclude this section (and conclude the description of our solution to the reverse Skolemization problem) by presenting the examples of the previous sections and their solutions using our final method.

Example 3.1

$$\mathbf{H} = \{ \forall X\, ontable(X) \}, \quad (1)$$

$$\mathbf{F} = \{ g \leftarrow \exists Y (red(Y) \wedge ontable(Y)), \quad (2)$$

$$\exists Z\, red(Z), \quad (3)$$

$$red(a), \quad (4)$$

$$\neg ontable(a) \}. \quad (5)$$

$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\},$ (1′)

$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{red(Y)} \wedge \mathtt{ontable(Y)},$ (2′)

$\mathtt{red(c)} \leftarrow \mathbf{F} \not\models \exists \mathtt{X}\,(\mathtt{red(X)} \wedge \mathtt{X} \in \mathit{NSHU}),$ (3′)

$\mathtt{red(a)},$ (4′)

$\neg\,\mathtt{ontable(a)} \,\}$ (5′)

We cannot explain $\mathtt{g}$ with $\mathtt{Y} = \mathtt{a}$ due to (5′), and the antecedent of (3′) is not satisfied since

$$\mathbf{F} \models (\mathtt{red(a)} \wedge \mathtt{a} \in \mathit{NSHU}).$$

There is therefore, correctly, no explanation of $\mathtt{g}$.

Example 3.2

$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\},$ (1′)

$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{red(Y)} \wedge \mathtt{ontable(Y)},$ (2′)

$\mathtt{red(c)} \leftarrow \mathbf{F} \not\models \exists \mathtt{X}\,(\mathtt{red(X)} \wedge \mathtt{X} \in \mathit{NSHU}),$ (3′)

$\neg\,\mathtt{red(a)},$ (4′)

$\neg\,\mathtt{ontable(a)} \,\}$ (5′)

Here we correctly explain $\mathtt{g}$ with theory $\{\, \mathtt{ontable(c)} \,\}$ since the antecedent of (3′) is true (we cannot show that $\mathtt{c}$'s definition is satisfied by another individual) and we cannot show the assumption simply inconsistent.

Example 3.3

$\mathbf{H} = \{\, \forall \mathtt{X}\, \mathtt{ontable(X)} \,\},$ (1)

$\mathbf{F} = \{\, \mathtt{g} \leftarrow \forall \mathtt{X}\, \mathtt{ontable(X)},$ (2)

$\neg\,\mathtt{ontable(a)} \,\},$ (3)

$\mathbf{H}' = \{\, \mathtt{ontable(X)} \,\},$ (1′)

$\mathbf{F}' = \{\, \mathtt{g} \leftarrow \mathtt{ontable(c)} \wedge \mathbf{F} \not\models \exists \mathtt{X}\,(\neg\,\mathtt{ontable(X)} \wedge \mathtt{X} \in \mathit{NSHU}),$ (2′)

$\neg\,\mathtt{ontable(a)} \,\},$ (3′)

First note that (2′) comes from applying the identity

$$(p \leftarrow q) \leftarrow r \equiv p \leftarrow (q \wedge r)$$

to the standard form of our solution.

Then in this example the hypothesis $\texttt{ontable(c)}$ is simply consistent but the minimality condition is not satisfied since we have

$$\mathbf{F} \models (\neg \texttt{ontable(a)} \wedge \texttt{a} \in \mathit{NSHU})$$

which rules out using (2′) to explain $\texttt{g}$. There is, correctly, no other explanation of $\texttt{g}$.

Example 4.1

$\mathbf{H} = \{ \forall \texttt{X} \forall \texttt{Y} \neg \texttt{on(X,Y)} \},$ (1)

$\mathbf{F} = \{ \forall \texttt{X} (\texttt{cleartop(X)} \leftarrow \neg \exists \texttt{Y on(Y,X)}),$ (2)

$\texttt{on(a,b)} \}.$ (3)

$\mathbf{H}' = \{ \neg \texttt{on(X,Y)} \},$ (1′)

$\mathbf{F}' = \{ \texttt{cleartop(X)} \leftarrow \neg \texttt{on(f(X),X)} \wedge \mathbf{F} \not\models \exists \texttt{Y} (\texttt{on(Y,X)} \wedge \texttt{Y} \in \mathit{NSHU}),$ (2′)

$\texttt{on(a,b)} \}.$ (3′)

We can explain $\texttt{cleartop(a)}$ with theory $\{ \neg \texttt{on(f(a),a)} \}$ since it is simply consistent and we can find no other block that is on $\texttt{a}$. For goal $\texttt{cleartop(b)}$ however, we have that

$$\mathbf{F} \models (\texttt{on(a,b)} \wedge \texttt{a} \in \mathit{NSHU})$$

thus ruling out the the hypothesis $\neg \texttt{on(f(b),b)}$ and preventing any explanation.

Example 4.2

$\mathbf{H} = \{ \forall \texttt{X ontable(X)} \},$ (1)

$\mathbf{F} = \{ \texttt{red(a)},$ (2)

$\exists \texttt{X red(X)} \}.$ (3)

$\mathbf{H}' = \{ \texttt{ontable(X)} \},$ (1′)

$\mathbf{F}' = \{ \texttt{red(a)},$ (2′)

$\texttt{red(c)} \leftarrow \mathbf{F} \not\models \exists \texttt{X} (\texttt{red(X)} \wedge \texttt{X} \in \mathit{NSHU}) \}.$ (3′)

Clearly we will only get the desired theory

$$\{ \texttt{ontable(a)} \}$$

to explain the query

$$\texttt{?} \leftarrow \exists \texttt{X red(X)} \wedge \texttt{ontable(X)}$$

since the unwanted theory

$$\{\,\texttt{ontable(c)}\,\}$$

is ruled out as the antecedent of (3′) is not satisfied since

$$\mathbf{F} \models (\texttt{red(a)} \wedge \texttt{a} \in \mathit{NSHU}).$$

Example 4.3

$$\mathbf{H} = \{\, \forall \texttt{X}\ \texttt{ontable(X)}\,\}, \tag{1}$$

$$\mathbf{F} = \{\, \texttt{g} \leftarrow \exists \texttt{X}\ (\texttt{red(X)} \wedge \texttt{ontable(X)}), \tag{2}$$

$$\exists \texttt{X}\ (\texttt{red(X)} \wedge \neg \texttt{ontable(X)})\,\} \tag{3}$$

$$\mathbf{H}' = \{\, \texttt{ontable(X)}\,\}, \tag{1′}$$

$$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(X)} \wedge \texttt{ontable(X)}, \tag{2′}$$

$$\texttt{red(c)} \leftarrow \mathbf{F} \not\models \exists \texttt{X}\,(\texttt{red(X)} \wedge \neg \texttt{ontable(X)} \wedge \texttt{X} \in \mathit{NSHU}), \tag{3′}$$

$$\neg \texttt{ontable(c)} \leftarrow \mathbf{F} \not\models \exists \texttt{X}\,(\texttt{red(X)} \wedge \neg \texttt{ontable(X)} \wedge \texttt{X} \in \mathit{NSHU})\,\} \tag{4′}$$

Finally, in this example, the antecedents of (3′) and (4′) are satisfied as there are no other blocks. We try to assume $\texttt{ontable(c)}$ to explain $\texttt{g}$ but it is simply inconsistent due to (4′). Thus we cannot explain $\texttt{g}$, correctly.

4.5. Poole's Examples

For completeness, in this section we present the examples from [Poo86a] which inspired this work. Note that his Examples 1, 2, and 5 are not what we are concerned with, his Example 3 (solved in his Example 6) is almost exactly our Example 4.1, and his Examples 7A and 7B are our Examples 3.1 and 3.2.

Poole's Example 4 (solved in his Example 7)

$$\mathbf{H} = \{\, \forall \texttt{X}\ \texttt{ontable(X)}\,\}, \tag{1}$$

$$\mathbf{F} = \{\, \texttt{g} \leftarrow \exists \texttt{Y}\ (\texttt{red(Y)} \wedge \texttt{ontable(Y)}), \tag{2}$$

$$\exists \texttt{X}\ \texttt{red(X)}\,\} \tag{3}$$

$$\mathbf{H}' = \{\, \texttt{ontable(X)}\,\}, \tag{1′}$$

$$\mathbf{F}' = \{\, \texttt{g} \leftarrow \texttt{red(Y)} \wedge \texttt{ontable(Y)}, \tag{2′}$$

$$\texttt{red(c)} \leftarrow \mathbf{F} \not\models \exists \texttt{X}\,(\texttt{red(X)} \wedge \neg \texttt{ontable(X)} \wedge \texttt{X} \in \mathit{NSHU}), \tag{3′}$$

This example is very similar to our Example 4.2. We want to explain g with theory $\{ontable(c)\}$ since there are no non-Skolem individuals satisfying the definition of c. The proof is illustrated in Figure 4.1.

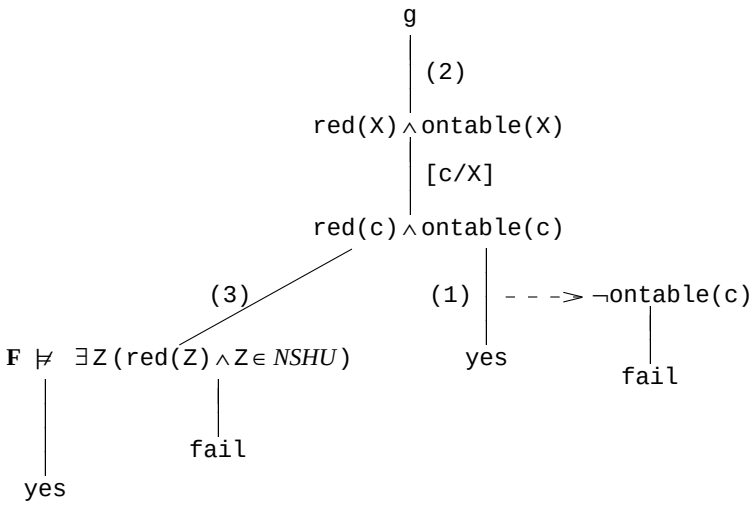

Figure 4.1 Poole's Example 4

Poole's Example 8a

$$\mathbf{H} = \{\forall X \forall Y \neg on(X,Y)\}, \tag{1}$$

$$\mathbf{F} = \{\forall X\,(cleartop(X) \leftarrow \neg \exists Y\, on(Y,X)), \tag{2}$$

$$on(a,b), \tag{3}$$

$$red(b), \tag{4}$$

$$g_1 \leftarrow \forall X\, cleartop(X)\}. \tag{5}$$

$$\mathbf{H}' = \{\neg on(X,Y)\}, \tag{1'}$$

$$\mathbf{F}' = \{cleartop(X) \leftarrow$$

$$\neg on(f(X),X) \wedge \mathbf{F} \nvdash \exists Y(on(Y,X) \wedge Y \in NSHU), \tag{2'}$$

$$on(a,b), \tag{3'}$$

$$red(a), \tag{4'}$$

$$g_1 \leftarrow cleartop(c_1) \wedge$$

$$\mathbf{F} \nvdash \exists Z(\neg cleartop(Z) \wedge Z \in NSHU)\}. \tag{5'}$$

Intuitively, we should not be able to explain g_1 since there is a block (b) whose top is not

clear. The proof is shown in Figure 4.2.

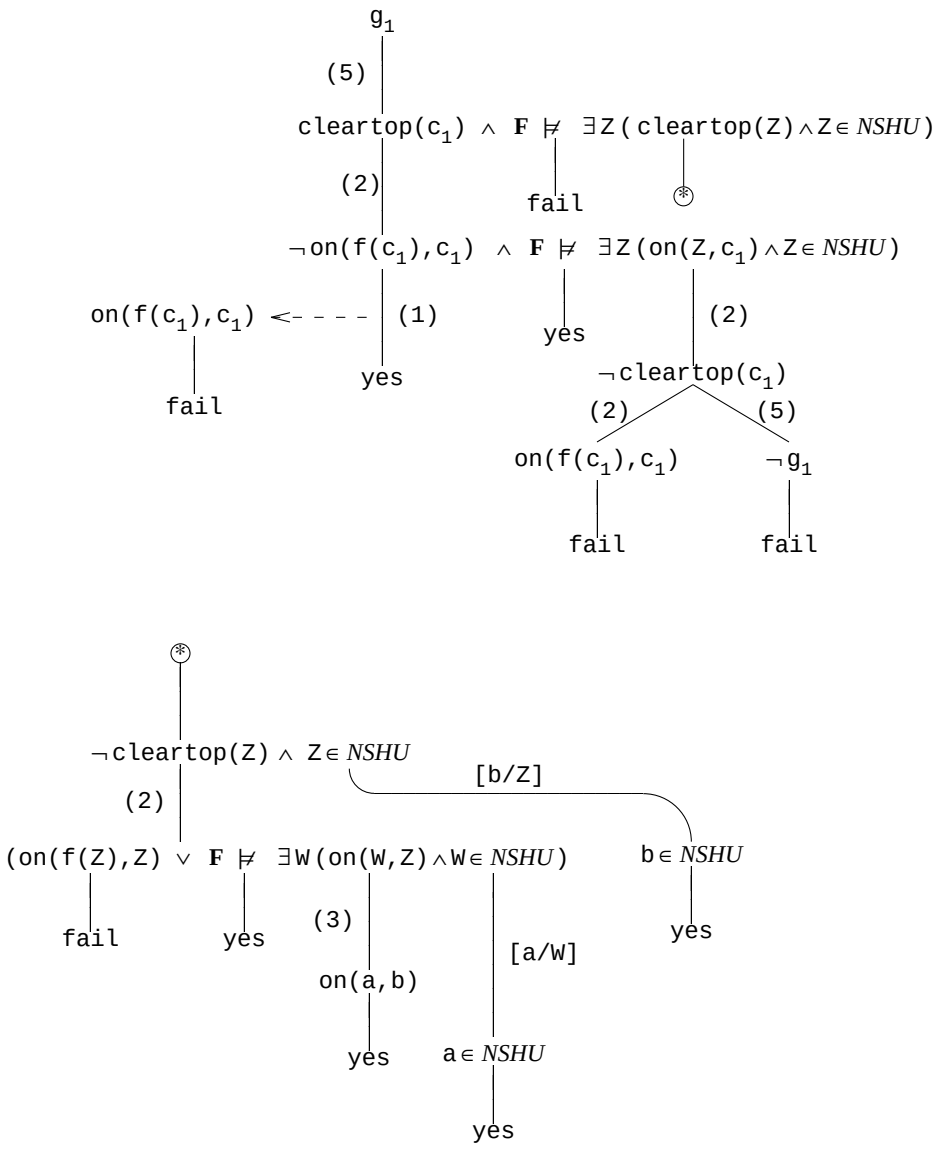

Figure 4.2 Poole's Example 8a

Poole's Example 8b

$$\mathbf{H} = \{\, \forall X \forall Y \neg \mathtt{on}(X,Y) \,\}, \tag{1}$$

$$\mathbf{F} = \{\, \forall X\, (\mathtt{cleartop}(X) \leftarrow \neg \exists Y\, \mathtt{on}(Y,X)), \tag{2}$$

$$\mathtt{on}(a,b), \tag{3}$$

$$\mathtt{red}(b), \tag{4}$$

$$g_2 \leftarrow \forall X\, (\mathtt{cleartop}(X) \leftarrow \mathtt{red}(X))\,\}. \tag{5}$$

$$\mathbf{H}' = \{\neg \mathtt{on(X,Y)}\}, \tag{1′}$$

$$\mathbf{F}' = \{\mathtt{cleartop(X)} \leftarrow$$

$$\neg \mathtt{on(f(X),X)} \wedge \mathbf{F} \nvDash \exists \mathtt{Y}(\mathtt{on(Y,X)} \wedge \mathtt{Y} \in \mathit{NSHU}), \tag{2′}$$

$$\mathtt{on(a,b)}, \tag{3′}$$

$$\mathtt{red(a)}, \tag{4′}$$

$$\mathtt{g}_2 \leftarrow \mathtt{cleartop}(\mathtt{c}_2) \wedge$$

$$\mathbf{F} \nvDash \exists \mathtt{Z}(\neg \mathtt{cleartop(Z)} \wedge \mathtt{red(Z)} \wedge \mathtt{Z} \in \mathtt{NSHU})\}. \tag{5′}$$

$$\mathtt{g}_2 \leftarrow \mathtt{red}(\mathtt{c}_2) \wedge$$

$$\mathbf{F} \nvDash \exists \mathtt{Z}(\neg \mathtt{cleartop(Z)} \wedge \mathtt{red(Z)} \wedge \mathtt{Z} \in \mathtt{NSHU})\}. \tag{6′}$$

This time we want to be able to explain $\mathtt{g}_2$ since the only block which does not have a clear top is red, and we are only trying to show that all non-red blocks have clear tops. The proof is illustrated in Figure 4.3. The dotted branch is identical to the top part of Figure 4.2 except with $\mathtt{c}_2$ rather than $\mathtt{c}_1$.

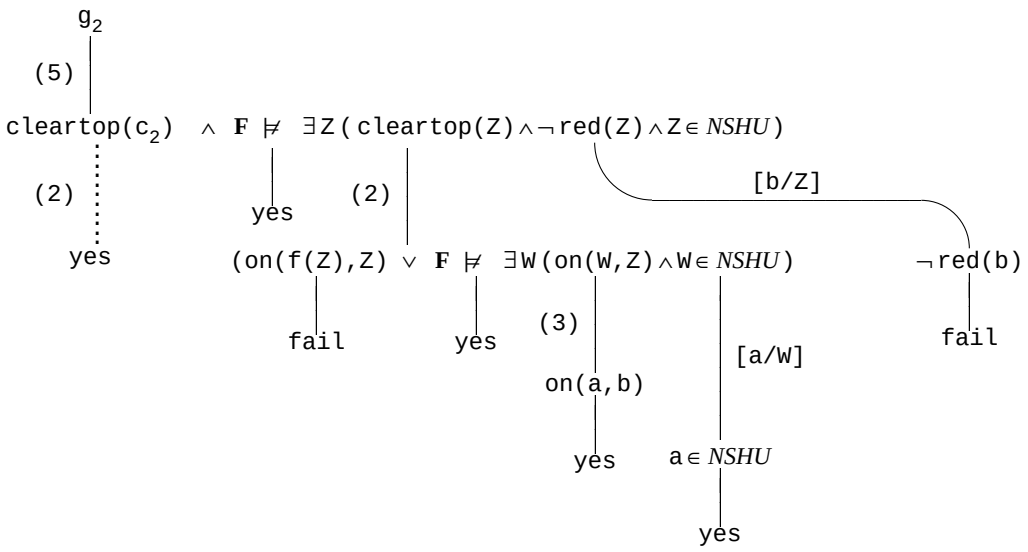

Figure 4.3 Poole's Example 8b

4.6. Interpretation vs. compilation

The solutions described above allow Theorist to *dynamically* compute overriding instances of the unique names hypothesis (Section 3.3) required to preserve the minimally attributive semantics of the original statements. In nonmonotonic reasoning systems such as Theorist, the question arises of how to effect the non-monotonicity and update the system's knowledge base in light of new input. This issue manifests itself within the scope of the RSP when existentially quantified statements are affected by the input, either directly or indirectly, as we shall see. We now consider a version of the previous solution which ''compiles in'' the intended interpretation and discuss its relationship to the previous, interpreted, alternative.

Suppose we have a Theorist database $(\mathbf{F}, \mathbf{H})$ to which we wish to add the fact

$$\forall \bar{Y} \exists X \, p(\bar{Y}, X).$$

We Skolemize to get

$$p(\bar{Y}, f(\bar{Y}))$$

where $p(X)$ is the defining sentence of the Skolem function f, *i.e.*

$$f = \varepsilon X . p(\bar{Y}, X).$$

In order to preserve the minimally attributive aspect of the original statement, we must

check whether any other individual in the database satisfies the definition of `f`. That is, we must check whether

$$\mathbf{F} \models \exists \mathtt{Y}\, p(\mathtt{Y}).$$

for any definite individual `Y`.

If we find an individual satisfying `f`'s definition, then we can simply leave the database unchanged with respect to the existence of the individual. For example, adding the statement

$$\exists \mathtt{X}\, \mathtt{red(X)}$$

to facts

$$\mathbf{F} = \{\, \mathtt{red(a)} \,\}$$

should add nothing to **F**.

If we don't find an individual satisfying `f`'s definition then, as before, we are justified in creating a new individual, which we do by simply adding the (Skolemized) clause

$$\mathtt{red(c)}$$

to **F**.

A more complicated situation arises when dealing with Skolem functions, as opposed to constants. In this case, we wish to ''compile-in'' the interpretation of a formula such as

$$\forall \mathtt{X} \exists \mathtt{Y}\, \mathtt{mother\_of(X,Y)}$$

which when converted to clauses becomes

$$\mathtt{mother\_of(X,m(X))}$$

where `m` is a Skolem function such that

$$\mathtt{m(X)} = \varepsilon \mathtt{Y.mother\_of(X,Y).}$$

Suppose we have an existing fact concerning a certain individual's mother, such as

$$\mathtt{mother\_of(pebbles,wilma).}$$

Now, in keeping with our desired minimal attributive interpretation of the existential statement, we would like to prevent the creation of an individual

$$\mathtt{m(pebbles)}$$

since we have that `wilma` satisfies the definition of `m`. However, we would accept as valid the use of

$$\texttt{mother\_of(wilma, m(wilma))}$$

since we cannot deduce any other individual which is the mother of `wilma`. The fact that we require to reflect the addition of the existential statement would then be of the form

$$\texttt{mother\_of(X, f(X))} \leftarrow \neg\, \texttt{eq(X, wilma).}$$

In general, the antecedent would list all those individuals which satisfied the definition of the Skolem term. This use of the homogeneous form [Cla78] might facilitate updating the database, depending on the implementation, particularly the implementation of equality.

It is not enough to perform this check only when adding existential statements to the database. When adding *any* new fact we must recheck *all* statements containing Skolem terms in case the new fact references an individual which now (after adding the fact) satisfies the definition of one of the Skolem terms. If this is the case, we can remove the fact containing the Skolem term. For example, adding

$$\texttt{red(a)}$$

to

$$\mathbf{F} = \{\, \exists \texttt{X red(X)} \,\}$$

should prevent the use of the Skolem term posited by the existential statement (after Skolemization).

Similarly, when deleting a fact we have to check all those sentences containing Skolem terms which we previously ‘‘ignored’’ due to the definitions of the terms being satisified by other (non-Skolem) terms. If the fact which is being deleted figured in the satisfaction of the Skolem term’s definition and we can no longer find an individual satisfying the definition, then we must admit the Skolem term as legitimately denoting a new individual. For example, removing

$$\texttt{red(a)}$$

from

$$\mathbf{F} = \{\, \exists \texttt{X red(X)}, \; \texttt{red(a)} \,\}$$

should ‘‘reintroduce’’ the Skolem individual.

The compilation of the intended interpretation of Skolem terms into the database during update comes at a price however. Since each search for satisfying individuals involves a test of validity, it is semi-decidable for any strong logic, just as Theorist's normal inference procedure is. Thus we are trading the cost (and potential non-termination) of inference at the meta-level during goal solution against a similar cost to be incurred as the database is updated. Whether we wish to incur this cost at ''update-time'' or at ''query-time'' becomes a matter of circumstance.

It may be that rather than the sort of ''exact'' manipulation of the database described above, we might want Theorist to interact with the user to determine his or her intentions. For example, upon detecting that an existential sentence was already satisfied by other individuals, the user might want to either change the existential statement to properly characterize the difference or might simply assert new facts about the individuals identified by Theorist. Similarly, upon deletion of sentences, the user might want to purge all references to satisfying individuals regardless of which facts they came from. Finally, when Skolem terms are admitted as legitimate, the user might want to give them descriptive names identifying the intended interpretation and making answers more comprehensible.

Chapter 5

Implementation

In this chapter we describe the details of our implementation of the modified Theorist interpreter. Since much of our work was directed towards implementing aspects of Theorist not directly concerned with the RSP, this chapter will have a somewhat broader scope than the rest of the dissertation.

All implementation was done in C-PROLOG on a VAX11/780, hence the name of our version: CTheorist. Previous versions of Theorist used Waterloo Unix Prolog or Quintus Prolog. CTheorist has been ported to Turbo-Prolog for the IBM-PC family, but the restrictions of that PROLOG forced extensive modifications to the code described here [Fit89]. Pseudo-PROLOG code presented here may not be exactly as in the implementation, although in many cases it is, modulo debugging and other extraneous code.

5.1. A first-order front end

5.1.1. Parser/tokenizer

In order to experiment with the Skolemization process, Theorist required a front-end capable of recognizing statements in the full first-order predicate calculus rather than than simply clauses as it had previously had. We implemented a parser/tokenizer module which accepts the Theorist command set described in Table 5.1. A *formula* is any formula of the first-order predicate calculus and an *atom* is a positive literal.

fact *formula*	Assert *formula* as a fact
hypothesis *formula*	Assert *formula* as a hypothesis
askable *atom*	Mark *atom* as askable
meta *atom*	Mark *atom* as meta-evaluable
explain *formula*	Get Theorist to explain *formula*
input *file-list*	Read commands from files
end	Exit Theorist
list *category-list*	List Theorist's database
clear *category-list*	Clear Theorist's database
help *topic-list*	Get help on Theorist topics

Table 5.1 Theorist command set

The tokenizer uses standard PROLOG techniques for lexical analysis using one-character lookahead. The parser uses PROLOG production rules with actions to convert infix input formulae to a prefix form. It converts a variety of input syntaxes for connectives and quantifiers to a standard set of operators. Details of certain commands such as **list** and **clear** are also handled and converted to a standard form.

5.1.2. Conversion to clauses

For commands which manipulate formulae (**fact**, **hypothesis**), the prefix form returned by the parser must be converted into clause form as described in §2.2. This is performed by seven recursively descending predicates, each of which implements one of the steps in the conversion. The result is a list representing a set of clauses (disjunctions) each of which is in prefix form. Skolem functions are automatically generated by the appropriate routine to replace existentially quantified variables.

5.1.3. Conversion to rules

Finally, each clause is converted to a PROLOG-like rule before assertion to accomodate Theorist's backward-chaining proof procedure. In each clause, one disjunct is chosen to be the head of the rule (consequent) and the others form the body (antecedent). All contrapositives of the rule are generated, thus the choice of head literal is irrelevant. The use of contrapositives allows non-Horn clauses to be expressed as rules: the general clause

$$p \vee q$$

becomes the pair of rules

$$p \leftarrow \neg q$$
$$q \leftarrow \neg p .$$

5.2. Theorist as a set of inference rules

It is useful to think of Theorist as a collection of inference rules for a first-order language. Thus the various Theorist commands correspond to different possible inference steps, and the CTheorist implementation reflects this. The theorem-proving heart of CTheorist is an interacting set of recursive clauses which define the `prove` predicate: `prove(G,A,H1,H2)` is true when `G` has been proven (explained, really) with ancestor goals `A` using a theory consisting of hypotheses `H1` and `H2`. The predicate `proveAll` simply attempts to `prove` each member of the list that it is given (the empty list being trivially proved).

5.2.1. MESON rules

The first pair of inference rules implement the MESON procedure which permits Theorist to use general clauses. Any formula asserted as a **`fact`** can be used to further a proof:

```
prove(G,A,H1,H2) :- fact(G :- B),
                    proveAll(B,A,H1,H2).
```

The list of ancestor goals is maintained to allow proof by contradiction if the current goal is the negation of an ancestor goal, as required by the MESON procedure [Lov78]:

```
prove(G,A,H1,H2) :- negate(G,NG),
                    member(G,A).
```

5.2.2. Meta rules

The second pair of inference rules implement Theorist's meta-commands: attempting to prove a predicate marked as **`askable`** results in query being posed of the user if the value cannot be deduced:

```
prove(G,A,H1,H2) :- askable(G),
                    answered(G),
                    answeredTrue(G).
prove(G,A,H1,H2) :- askable(G),
                    not answered(G),
                    answerTrue(G).
```

The predicates `answered` and `answeredTrue` check whether the predicate has been previously asked and what its value was; `answerTrue` interacts with the user and saves the answer. Note that the user can answer ‘‘true’’, ‘‘false’’, or ‘‘don’t know’’ to these queries.

Predicates marked as **`meta`** are evaluated by PROLOG:

```
prove(G,A,H1,H2) :- meta(G),
                    call(G).
```

5.2.3. Hypothetical rules

Finally, the theory formation aspect of Theorist is implemented by using formulae asserted as **`hypotheses`**:

```
prove(G,A,H1,[G|H2]) :- hypothesis(G :- B),
                        proveAll(B,[G|A],H1,H2),
                        negate(G,NG),
                        not prove(NG,A,H2,H2).
```

If the formula is not atomic, CTheorist attempts to prove its antecedent. 6. If succesful the consistency of the assumption is checked by making a recursive call to the `prove` predicate to attempt to show the negation of the hypothesis. If this fails then the hypothesis can be consistently assumed. In the current implementation, hypotheses are not permitted in the consistency check to rule out other hypotheses. The semantics of this restriction is part of current work in the Theorist project (*e.g.* [Goo89, SaG89]).

We should note that in the actual implementation, the `prove` predicate has several extra arguments which maintain state such as: the depth of the current proof for tracing; variable bindings for pretty-printing (since C-PROLOG does not provide this); and flags to indicate, for example, whether or not the current goal is part of a consistency check.

---

6. We could always make our hypotheses atomic by transforming then to **`hypothesis`** `newAtom` **`and fact`** `newAtom` ↔ *formula*, **similar to the naming transformation referred to in Footnote 2.**

5.3. Paramodulation

We augmented Theorist to handle the equality relation by implementing paramodulation [RoW69] as an additional inference rule. This was used as mentioned in Sections 4.1 and 4.2 to implement parts of the initial solutions and for related explorations of the utility of equality in general. The additional clause for `prove` is

```
prove(G,A,H1,H2) :- fact(eq(X,Y):-B),
                    paramodulate(eq(X,Y),G,Gp),
                    proveAll(B,[G|A],H1,H3),
                    prove(Gp,A,H3,H2).
```

The predicate `paramodulate(T1,T2,T3)` is true if the result of paramodulating *from* `T1` *into* `T2` is `T3`. The from literal is the statement of equality, for example suppose this is `eq(a,b)`. The into literal is our ‘‘source’’, say `p(a)`. Then the result of paramodulating from `eq(a,b)` into `p(a)` is `p(b)`.

The implementation of `paramodulate` is straightforward and consists of enumerating subterms of `T2` until one is found which unifies with either `X` or `Y` (`X` or `a` in the example). The other term of the equality (`Y` or `b` in this case) is then substituted for occurances of `X` in `T2` to yield `T3`. Note that this is not quite complete paramodulation: the specification of the inference rule [RoW69, p. 139] requires only that the two terms unify with a common term, not that they necessarily unify with each other. For simplicity of implementation we use the weaker version which obviates the need for a search for a common term (*i.e.*, we are ignoring the warning in [RoW69] that ‘‘viewing paramodulation as mere substitution of equals is like viewing resolution as mere pattern matching’’ [page 140]). A detailed study of the effects of this restriction and of the costs and benefits of deduction with equality in general, while useful and important, was beyond the scope of this work.

We made two refinements to the above naive implementation of paramodulation in the interests of improving performance:

a) We test that `G` and `Gp` are not identical after the paramodulation. While such a result would be correct and reflects the reflexivity of equality, it can only cause loops in Theorist’s

proof procedure. Resolution by itself provides the `eq(X,X)` axiom.

b) To further prevent loops, we maintain a list of *paramodulation ancestors* throughout a proof. That is, whenever we use the paramodulation inference rule we add the term which we paramodulated *into* to this list. The list is checked immediately after testing condition (a): if `Gp` is a member of it then we can safely fail the proof. 7. As above, such inferences would be correct (in this case they reflect the symmetry of equality) but again they can only lead to loops in an SLD-resolution prover like Theorist's.

5.4. Ehypotheses

In connection with the use of equality, second-level hypotheses for the implementation of the early solutions were added to the front end and are generated automatically during Skolemization. Extensive use is made of PROLOG meta-predicates for the construction of terms and clauses.

Use of these ehypotheses is implemented as yet another inference rule (clause for `prove`). This clause can only be used during consistency checks, but is otherwise similar to the clause governing the use of hypotheses. That clause is also modified to permit the use of the ehypotheses during the attempt to show the negation of the assumption.

The work spent developing this, while not completely correct with respect to the reverse Skolemization problem, nonetheless provided useful insight into the use of interacting levels of hypotheses with equality. For example, we believe that inheritance hierarchies with arbitrary exceptions can be modelled using hypotheses which are allowed to rule each other out. Criteria for deciding among multiple extensions is, as always in Theorist, theory preference information. In this case the required information amounts to indicating which hypotheses can rule out which. While work here was only rudimentary, we believe that explicit manipulation of statements and hypotheses concerning equality is the best approach to such hierarchical classification problems.

As another example, [Goe89] is pursuing equality as a means of describing and computing analogical information. In this case it is the statement of equality *itself* that must be assumed,

---

7. Note that we do not unify here; the terms must be literally identical.

under certain criteria. Again, theory preference knowledge can decide between competing hypotheses.

5.5. Meta-proofs

5.5.1. Theory formation solution

In accordance with the solution presented in Section 4.3, if we only want to rule out illegitimate theories, we modify the rule governing the use of hypotheses to check for Skolem terms in any hypothesis it is considering. If one is found then, in addition to the normal test for consistency, the hypothesis can be ruled out by showing that the Skolem term's definition is satisfied. The modified clause for `prove` is therefore:

```
prove(G,A,H1,[G|H2]) :- hypothesis(G:-B),
                        proveAll(B,[G|A],H1,H2),
                        not ruleOut(G,A,H2).

ruleOut(G,A,H) :- negate(G,NG),
                  prove(NG,A,H,H).

ruleOut(G,A,H) :- skolemTerm(G,Sk),
                  definition(Sk,Def),
                  prove(Def,A,H,H),
                  not skolemTerm(Def,_).
```

The predicate `skolemTerm(F,Sk)` is true if `Sk` is a Skolem subterm of formula `F`, and will enumerate all such terms by backtracking. It fails if there are no Skolem subterms in `F`, hence its use to check for elements of the NS-Herbrand universe (see Section 4.1). The predicate `definition(Sk,Def)` simply looks up the definition of `Sk` which was generated by the front end during Skolemization. The first clause for `ruleOut` performs simple consistency checking, the second attempts to prove that a Skolem term's definition is satisfied by a non-Skolem term.

5.5.2. General solution

Following the specification in Section 4.4, the front end converts input statements of the form

$$\forall \bar{Y} \exists X \; p(\bar{Y}, X)$$

to

$$p(\bar{Y}, f(\bar{Y})) \leftarrow \texttt{not}\,(\,\texttt{prove}(p(\bar{Y}, Z), A, H, H),\ \texttt{not}\,\texttt{SkolemTerm}(Z, \_)).$$

where `prove` is marked as **`meta`** and cannot use new hypotheses. In fact there are some technical details regarding variable bindings and the like which we have ignored here, but the idea and its implementation is hopefully clear enough.

Chapter 6

Conclusions

We have examined the role of existentially quantified statements in resolution-based deductive systems, and observed that the lack of precise semantics for the Skolem terms introduced mechanically during conversion to the required clausal form can lead to intuitively incorrect answers. This has been called the reverse Skolemization problem (RSP). We defined the concept of *minimal attribution* as the desired semantics for such terms in a hypothetical reasoning environment (and in general). We showed how minimal attribution makes intuitive sense, and have illustrated this with examples using the Theorist paradigm.

We examined how the RSP occurs with respect to two different ways Skolem terms can arise: *assertionally* and *verificationally*. In the verificational case, the Skolem term is really a *placeholder* for a universally quantified variable. Our solution ensured that Skolem terms appearing verificationally did not get used in theories where they were not treated as universally quantified variables. In the assertional case, the Skolem terms become *default rules* for naming individuals satisfying certain properties. Our solution ensured that the Skolem terms were only introduced when existing individuals did not satisfy these properties, thus preserving minimally attributive semantics.

We described a progession of solutions to the RSP, starting with the intuitively ‘‘easiest’’ solution which involved replacing Skolem terms with variables before attempting to show their negation. These variables are instantiated during consistency checking, and we test that the instantiating individuals belong to the non-Skolem Herbrand universe of our set of clauses. This was justified by examining the semantics of the original, non-clausal formulae.

The first solution was then modified to consider only individuals which also satisfied the definition of the Skolem term they were replacing. This prevented individuals which explicitly *could not* be identical to a Skolem term from ruling out an assumption concerning it.

Finally, we observed that, to preserve minimal attribution, *only* the satisfaction of the definition of a Skolem term by a non-Skolem term is required to rule out an assumption concerning the Skolem term. This was illustrated by examples where there was no negative information which would allow us to rule out a theory, but where introducing a Skolem term would violate minimal attribution. Our solution involved a meta-level proof of the necessity of the Skolem term and a simple consistency check of the assumption to ensure that all information about the Skolem term was considered.

We showed how the RSP can occur in deductive situations where our solution based on ruling out theories during consistency checking would obviously not apply. Our solution was shown to be readily adaptable to such situations and was shown to be capable of preserving minimally attributive semantics in these situations as well. Several examples illustrated the sufficiency of this final approach.

The meta-level computation done in order to enforce minimal attribution is semi-decidable for any strong logic, thus we described how minimal attribution can be enforced at *update-time* rather than at *query-time* (without avoiding the cost however). We described how, for a variety of user/machine interactions involving the addition and deletion of Skolem terms arising from existential statements, the meta-proof required for our *dynamic* solution can be used to maintain a *static* database. We also showed how all statements, not just those containing Skolem terms, can affect the minimally attributive semantics, and described possible ways of handling these updates.

We showed how our solutions could be easily incorporated into the Theorist framework, and provided detailed examples of an implementation of the dynamic version of our solution. We also described in some detail other aspects of the Theorist implementation which contributed to the exploration and understanding of the RSP and of the role of Skolem terms in general.

Future work could certainly focus on maintaining a ‘‘criterion of identity’’ in order to enforce minimal attribution more efficiently. As well, user-interface issues such as those discussed in Section 4.6, which are really a form of dynamic interaction to determine semantics,

could be further explored in the context of learning concepts. Since this work focuses on identity, further work on deductive systems with equality would no doubt prove useful. Indeed, equality is a powerful tool for the natural axiomatization of many domains and so deserves further attention. Since the costs of reasoning with equality are high, a careful examination of the benefits is required.

We feel that existential statements arise naturally in the axiomatization of many situations. In order to accomodate such statements in a resolution-based system, a precise semantics for the Skolem terms introduced during conversion to clauses must be decided upon and enforced by the system. For example, verificational Skolem terms are unavoidable if universally quantified formulae are to be permitted in rule antecedents, and such rules are often useful. As described above, assertional Skolem terms interpreted in a minimally attributive way become *default rules* for naming individuals by identifying their properties, another useful representational tool.

This dissertation has cleared the philosophical air surrounding Skolem terms, permitting unrestricted use of full clausal logic in deductive systems. While minimally attributive semantics are not the only possible way of interpreting existentially quantified statements, we believe that it is the most intuitively reasonable way. We have illustrated the natural connection between these semantics and hypothetical reasoning, thereby clearing the way to full use of existential statements in these systems also.

References

[BMN80] M. Bergmann, J. Moor and J. Nelson (1980), *The logic book*, Random House, New York, NY.

[Bow82] K. A. Bowen (1982), Programming with full first-order logic, *Machine Intelligence*, vol. 10, J. E. Hayes, D. Michie and Y. Pao (eds.), Ellis-Horwood, 421-440.

[Cam84] J. A. Campbell (1984, ed.), *Implementations of prolog*, Ellis-Horwood Publications, New York.

[ChL73] C. L. Chang and R. C. T. Lee (1973), *Symbolic logic and mechanical theorem proving*, Academic Press, Inc., New York, NY.

[Cla78] K. L. Clark (1978), Negation as failure, *Logic and Data Bases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 293-322.

[Col83] A. Colmerauer (1983), Prolog in 10 figures, *Proceedings of Eight International Joint Conference on Artificial Intelligence (IJCAI-83)*, August 8-12, Karlsruhe, Germany, 487-499.

[CoP84] P. T. Cox and T. Pietrzykowski (1984), A Complete, Nonredundant Algorithm for Reversed Skolemization, *Theoretical Computer Science* **28**, 239-261.

[Fit89] P. Fitzsimmons (1989), The practical feasibility of Theorist: turf grass disease diagnosis, M.Sc. dissertation, Department of Computing Science, University of Alberta, Edmonton, Alberta, September [expected].

[Fuc88] K. Fuchi (1988), A springboard to the final stage of the FGCS project, *ICOT Journal*(20), June, 201-215.

[Gal86] J. H. Gallier (1986), *Logic for computer science*, Harper & Row, Publishers, Inc., New York, NY.

[GeN87] M. Genesereth and N. J. Nilsson (1987), *Logical foundations of artificial intelligence*, Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Goe85] R. Goebel (1985), A logic data model for the machine representation of knowledge, Ph.D. dissertation, Department of Computer Science, University of British Columbia, Vancouver, BC., November.

[GoG87] R. Goebel and S. D. Goodwin (1987), Applying theory formation to the planning problem, *Proceedings of the 1987 Workshop: The Frame Problem in Artificial Intelligence*, April 12-15, Morgan Kaufmann, Los Altos, CA, F. M. Brown (ed.), 207-232.

[Goe89] R. Goebel (1989), A sketch of analogy as reasoning with equality hypotheses, Department of Computing Science, University of Alberta, Edmonton, March, 17 pages.

[Goo87] S. D. Goodwin (1987), Representing frame axioms as defaults, M.Math. dissertation, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 186 pages.

[Goo89] S. D. Goodwin (1989), Semantic tools for the analysis and synthesis of a vocabulary for hypothetical reasoning, Ph.D. dissertation, Department of Computer Science, University of Alberta , Edmonton, Alberta, 45 pages [proposal].

[HaM86] S. Hanks and D. McDermott (1986), Default reasoning, nonmonotonic logic, and the frame problem, *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, August 11-15, University of Pennsylvania, Philadelphia, Pennsylvania, 328-333.

[ICO88] ICOT (1988, ed.), *Proceedings of the International Conference on Fifth-generation Computer Systems*, Nov. 28 - Dec. 3, Tokyo, Japan [ISBN4-274-07463-3].

[Jac86] W. K. Jackson (1986), A theory formation framework for learning by analogy, M.Math dissertation, Department of Computer Science, University of Waterloo, Waterloo, Ontario, December, 88 pages.

[JoP85] M. Jones and D. Poole (1985), An expert system for educational diagnosis based on default logic, *Proceedings of the Fifth International Workshop on Expert Systems and their Applications*, May 13-15, Palais des Papes, Avignon, France, 573-583.

[Kir87] B. Kirby (1987), Preferring the Most Specific Extension (in *Experiments in the Theorist paradigm: A collection of student papers on the Theorist project*), Research Report CS-87-30, Department of Computer Science, University of Waterloo, May.

[Kor83] W. F. Kornfeld (1983), Equality for prolog, *Proceedings of Eight International Joint Conference on Artificial Intelligence (IJCAI-83)*, August 8-12, Karlsruhe, Germany, 514-519.

[Kow74] R. A. Kowalski (1974), Predicate logic as a programming language, *Information Processing 74*, 569-574.

[Lei69] A. C. Leisenring (1969), *Mathematical logic and Hilbert's* ε*-symbol*, MacDonald Technical & Scientific, London, England.

[LiG89] D. Lin and R. Goebel (1989), Computing circumscription of ground theories with Theorist, Department of Computer Science, University of Alberta, Edmonton, AB.

[Llo87] J. W. Lloyd (1987), *Foundations of logic programming*, edition 2, Springer-Verlag, Berlin.

[Lov78] D. W. Loveland (1978), *Automated theorem proving: A logical basis*, North-Holland Publishing, New York.

[McC80] J. McCarthy (1980), Circumscription—a form of non-monotonic reasoning, *Artificial Intelligence* **13**(1&2), 27-39.

[Nil80] N. J. Nilsson (1980), *Principles of artificial intelligence*, Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Poo85] D. L. Poole (1985), On the comparison of theories: Preferring the most specific explanation, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-85)*, August 18-23, UCLA, Los Angeles, CA, A. Joshi (ed.), 144-147.

[Poo86a] D. L. Poole (1986), Variables in hypotheses, Research report CS-86-44, Department of Computer Science, University of Waterloo, Waterloo, Ontario, September.

[Poo86b] D. L. Poole (1986), Default reasoning and diagnosis as theory formation, Research Report CS-86-08, Department of Computer Science, University of waterloo, March.

[PGA87] D. L. Poole, R. Goebel and R. Aleliunas (1987), Theorist: a logical reasoning system for defaults and diagnosis, *The Knowledge Frontier: Essays in the Representation of Knowledge*, N. J. Cercone and G. McCalla (eds.), Springer-Verlag, New York, 331-352.

[Poo88a] D. Poole (1988), A logical framework for default reasoning, *Artificial Intelligence* **36**(1), 27-48.

[Poo88b] D. L. Poole (1988), Default and abductive reasoning: An architecture for explanation and prediction, *Journal of Intelligent Systems* [submitted].

[Poo89] D. L. Poole (1989), What the lottery paradox tells us about default reasoning, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, May 15-18, Toronto, ON, R. J. Brachman, H. J. Levesque and R. Reiter (eds.), 333-340.

[Rei80] R. Reiter (1980), A logic for default reasoning, *Artificial Intelligence* **13**(1&2), 81-132.

[Rei81] R. Reiter (1981), On closed world data bases, *Readings in artificial intelligence*, B. L. Webber and N. J. Nilsson (eds.), Morgan Kaufmann, Los Altos, CA, 199-240 [Also in *Logic and data bases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 1978, pages 55-76].

[Rob65] J. A. Robinson (1965), A machine-oriented logic based on the resolution principle, *ACM Journal* **12**(1), January, 23-41.

[RoW69] G. Robinson and L. Wos (1969), Paramodulation and theorem-proving in first-order theories with equality, *Machine Intelligence*, vol. 4, B. Meltzer and D. Michie (eds.), American Elsevier Publishing Co., Inc., 135-150.

[SaG89] A. Sattar and R. Goebel (1989), Using crucial literals to select better theories, Department of Computing Science, University of Alberta, Edmonton, Alberta.

[Sti88] M. E. Stickel (1988), A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler, *Journal of Automated Reasoning* **4**(4), 353-380.

[THT87] D. S. Touretsky, J. Horty and R. Thomason (1987), A clash of intuitions: The current state of nonmonotonic reasoning, *Proceedings of Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, August 23-28, Milan, Italy, 476-482.

[Tub86] J. B. Tubman (1986), An expert system for educational diagnosis using Theorist, Research report CS-86-32, Department of Computer Science, University of Waterloo, Waterloo, Ontario, February, 93 pages.

[UmP85] Z. D. Umrigar and V. Pitchumani (1985), An experiment in programming with full first-order logic, *IEEE 1985 Symposium on Logic Programming*, July 15-18, Boston, Massachusetts, 40-47.

[vaK76] M. H. van Emden and R. A. Kowalski (1976), The semantics of predicate logic as a programming language, *ACM Journal* **23**(4), 733-742.

[vaG86] M. H. van Emden and R. Goebel (1986), Research at Waterloo on logic-mediated, knowledge-based, personal information systems, *Canadian Artificial Intelligence* **8**, 26-29.

[Yuk87] K. Yukawa (1987), Amalgamating functional and relational programming through the use of equality axioms, Ph.D. dissertation, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 149 pages.