Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A FORMAL LANGUAGE FOR REPRESENTATION OF AND REASONING ABOUT INDIRECT CONTEXT

Bijan Arbab

June 1988
CSD-880045

UNIVERSITY OF CALIFORNIA

Los Angeles

A Formal Language for Representation of and Reasoning about Indirect Context

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Computer Science

by

Bijan Arbab

1988

The dissertation of Bijan Arbab is approved.

[illegible]

Richard R. Muntz

[illegible]

Michel A. Melkanoff

[illegible]

D. Anthony Martin

Alonzo Church

Alonzo Church

[illegible]

D. Stott Parker, Committee Chair

University of California, Los Angeles

1988

TABLE OF CONTENTS

1 **Introduction** 1
  1.1 The Paradox of the Name Relation 2
  1.2 Reasons for Eliminating the Paradox 7
  1.3 Previous Solution 10
  1.4 Proposed Soution 11

2 **Alternative Interpretations of Equality** 13
  2.1 Alternative 2 14
  2.2 Alternative 1 18
  2.3 Alternative 0 19

3 **Proposition Surrogates** 22
  3.1 Development of Proposition Surrogates 23
  3.2 Proposition Surrogates Under Alt(1) and Alt(0) 27
  3.3 Algorithm for Obtaining Proposition Surrogates 28
  3.4 The Problem of Proposition Surrogates 32

4 **Proposition Surrogates and Pointers** 34
  4.1 Pointers v.s. Metalinguistic Devices 37
  4.2 New Algorithm for Obtaining Proposition Surrogates . . 40

5 **Formal Language** 45
  5.1 Primitive Symbols 47
  5.2 Formation Rules 48
  5.3 Rules of Inference 48
  5.4 Abbreviations 50
  5.5 Axioms 52

6 **Examples** 56
  6.1 Touchy Yacht Owner 56
  6.2 The Art Collector 57
  6.3 The Wise Men 59

7 **Implementation** 66

8 **Conclusion** 73

A Contextual Descriptions 76

B McCarthy's Solution 79

C Prolog Code 86

References 96

ACKNOWLEDGMENTS

I am grateful to my mentor, Professor Alonzo Church, for guidance and in depth comments on this dissertation. In particular, his discussions and objections, raised so industriously, contributed to the solution as presented.

I am indebted to Professor Stott Parker for many contributions, both in content and form, to this dissertation. He provided the insight needed to help me through many hard corners. It is unreasonable to ask for a more patient and careful mentor.

I am thankful to Professor Donald Michie for review of a preliminary draft of this work. He encouraged me to continue the work, when it was nothing more than some hastily made notes and half backed ideas.

The long term help and support provided by Dr. Farhad Arbab, Professor Daniel Berry, and Professor Michel Melkanoff.

IBM Scientific Center not only for taking me off of a strict diet but also for providing a very nice research environment.

Last but certainly not least, my family.

VITA

Published Papers

1. Bijan Arbab and Daniel Berry, *Operational and Denotational Semantics of Prolog,* Journal of logic programming, Vol. 4 No. 4, pp. 309-329, December 1987

2. Bijan Arbab, *Object Identification from Parallel Light Strip,* International Joint Conference on Artificial Intelligence (IJCAI-87), Milano, Italy, pp. 1145-1148, August 1987

3. Bijan Arbab, *MASK: An Object Identification Algorithm,* Spatial Reasoning and Multi-Sensor Fusion, pp. 107-117, Oct 1987

4. Bijan Arbab *The Use of Examples as Concept Specification Languages,* IBM Los Angeles Scientific Center Report, G320-2804, February 1987

5. Bijan Arbab, *Compiling Circular Attribute Grammars into Prolog,* IBM Journal of Research and Development, Vol 30, Num 3, pp. 294-309, May 1986

6. Bijan Arbab and Ivan Bratko, *A Problem and its Solution with regards to Warplan,* IBM Los Angeles Scientific Center Report, G320-2779, February 1986

7. Bijan Arbab and Donald Michie, *Synthesis of Human Understandable and Efficient Rules,* Machine Intelligence 11 (eds, D. Michie, Jean Hayes and Judith Richards), Turing Institute, Scotland, May 1985

8. Bijan Arbab and Donald Michie, *Generating Rules From Examples,* Proceeding of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85), Los Angeles, pp. 631-633, August 1985

9. Bijan Arbab, *Building Expert Systems by Generating Rules from Examples,* IBM Los Angeles Scientific Center Report, G320-2763, March 1985

10. Bijan Arbab, *Rule Generator User Manual,* IBM Los Angeles Scientific Center Report, G320-2762, February 1985

ABSTRACT OF THE DISSERTATION

A Formal Language for Representation of and Reasoning about Indirect Context

by

Bijan Arbab

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1988

Professor D. Stott Parker, Chair

The purpose of this dissertation is to present a method for elimination of a paradox from formal languages such as computer programming languages. Counterintuitive conclusions which may arise once certain type of statements are formalized are called paradoxical. These statements involve words like *search*, *think*, *seek*, *know*, *want*, *believe*, etc. The reasons for the desire to free formal languages from this paradox are both theoretical and pragmatic.

The paradox was originally discussed by Frege (1892) who pointed out that the fundamental problem involves the idea of sameness (identity). Particularly, if identity is a relation, then is it a relation between objects or between names of objects? The theoretical and philosophical issues regarding the paradox have been discussed by many, including Russell (1905), Whitehead and Russell (1910), Church (1950) (1951) (1983) (1986), and Ajdukiewicz (1960) (1967a) (1967b). All of these works are reviewed here. The practical reasons for elimination of the paradox have to do with formally representing facts about knowledge, belief, etc. Such a formal language should allow reasoning with statements about knowledge and belief. A paradoxical formal language allows derivation of useless, possibly contradictory, conclusions, making it harder to find valid and useful conclusions.

The method for the elimination of the paradox presented in this dissertation is a modification of an approach originally pointed out by Ajdukiewicz (1960) and Church (1983). One advantage of this approach is that it can be added to many formal languages without modifying the underlying logic. The modified proposition surrogates are then added to Church's (1940) simple theory of types and the computer programming language Prolog. This dissertation, however, is not concerned with specifying a particular set of suitable axioms for knowledge, belief or other such words which introduce an *indirect context*. The purpose is, rather, to conservatively extend a formal language such that it avoids certain problems arising in representing statements about knowledge, belief etc and makes it possible to state desired set of axioms about knowledge, belief, etc. Examples as to how statements of the above type can be formalized in the modified languages are presented.

CHAPTER 1

Introduction

The identity relation (=) and substitution of equals for equals are among the most primitive and intuitive ideas of formal languages. The fact that all of the ordinary propositional connectives can be defined in terms of equality by means of quantified variables is due to Tarski (1923). The fact that classical quantifiers themselves can be defined in terms of equality, with the help of λ, is due to Quine (1956) and Henkin (1963).

The substitutivity of identity, though simple, poses some challenging problems. For example, consider the historical fact that *Heinrich Schliemann made a search for the site of Troy* and also that Troy is located four miles from the mouth of the Dardanelles, that is: *Site of Troy = the location four miles from the mouth of Dardanelles.* Now, according to the rule of substitutivity of identity it can be deduced that *Heinrich Schliemann made a search for the location four miles from the mouth of Dardanelles.* The first sentence is true, and the last is false, which means that they are not equivalent, although the last sentence is obtained from the first by substituting for *the site of Troy* the equivalent expression *the location which is four miles from the mouth of Dardanelles.* As another example consider the sentence *Newton knew that 8 > 5* and the identity *8 = the atomic* *number of oxygen.* According to the substitution of equals for equals it can be concluded that *Newton knew that the atomic number of oxygen* > 5 which is not true by any stretch of the imagination.

These are a few examples, among many, of the so-called paradox of the name relation. The paradox of the name relation refers to counter intuitive conclusions that may arise as a result of application of the rule of substitution of equals for equals to sentences that are in the so-called indirect context. An indirect context is introduced by words such as *know, believe, searched, necessary, want, etc.* For example, even though *it is necessary that* $\sqrt{81} = 9$ and that $\sqrt{81} =$ *the number of planets,* it is not true that *it is necessary that the number of planets* $= 9$.

The thesis defended by this dissertation is that the identity relation and the rule of substitutivity of identity can be maintained in a formal language, with full force, without admitting paradoxical conclusions. Such a formal language can play an important role in studying various epistemological problems as well as addressing certain issues in knowledge representation.

1.1 The Paradox of the Name Relation

A historical study of formal languages points out not only the source of the problem illustrated in the preceding paragraphs, but also the key to its solution. The remaining of this section then is devoted to such a review. Additionally a detailed discussion of the reasons for the desire to eliminate the paradoxical conclusions and an outline of the solution is presented.

The predicate calculus and quantification theory was presented to the world in complete form by Frege (1879). This work presented truth-functional propositional calculus and analysis of propositions with function and arguments instead of subjects and predicates. Additionally, Frege introduced a system of logic in which derivations are carried out exclusively according to the form of the expressions. The rules for carrying out these derivations (rules of inference, rules of symbol manipulation) are clearly stated, as they must be for any formal language. Among these rules of inference is one which states that if a and b are the same, then a can be replaced anywhere by b and conversely. This rule of inference is known as substitutivity of identity or substitution of equals for equals. Frege (1879) took the idea of sameness to be a relation. The word sameness is used in the sense of identity and $a = b$ is used in the sense of *a is the same as b* or *a and b coincide*. In fact, sameness was taken to be a relation between names or signs of objects as opposed to objects. Frege was not completely satisfied with this choice, the reasons for which he described in a paper published in 1892.

Frege (1892) thus advocated a different approach to the identity relation and also argued against taking sameness as just a relation between objects or names of objects. Frege (1892) showed that if identity is taken only as a relation between the objects named by a and b then $a = b$ and $a = a$ would not seem different if $a = b$ is true.

Skolem (1923) used the following example to illustrate how the propositions expressed by $a = b$ and $a = a$ may differ. Consider the case of a man who has two proper names a and b. Suppose that some information about a is given, for example that a has five children. On another occasion b is introduced and it is explained that b is a (i.e., $b = a$). The proposition expressed by $b = a$ then contains information about b, namely that b has five children, because of prior knowledge about a. The proposition expressed by *b is a* ($b = a$) is therefore entirely different from the proposition expressed by *a is a* ($a = a$) and *b is b* ($b = b$). The latter two are completely trivial, but the former is not if something is known in advance about a but not b, or about b but not a.

In order to explain the difference in the propositions expressed by $a = b$ and $a = a$, Frege (1892) developed the idea that every name (sign, combination of words, sentence) has a denotation as well as what is called a sense (meaning, connotation). This is then used to explain the difference between propositions expressed by $a = b$ and $a = a$, even if $a = b$ is true. Accordingly, if $a = b$ is true, then the denotations of a and of b are indeed the same and also the truth-value of $a = b$ is the same as that of $a = a$. Nevertheless, the sense of a may differ from the sense of b. The proposition expressed by $a = b$ then may differ from the proposition expressed by $a = a$ in which case the two sentences do not express the same proposition.

The possible distinction between the senses of a and b is rooted in the different ways the denoted objects are given. Frege (1892) uses the following example to illustrate this distinction. Let a, b, c be straight lines connecting the corners of a triangle with the midpoint of the opposite sides. The names *intersection of a* *and b* and *intersection of b and c* denote the same point. However, the manner in which these points are indicated is different. Frege (1892) therefore, assigns a denotation (designated object, nominatum) as well as a sense (connotation, meaning), in which the manner of presentation is contained to every name (sign, sentences). For example, the denotation of the names *intersection of a and b*, *intersection of b and c* would be the same, but not their senses (the manner in which the denoted point is given).

Frege (1892) identifies three distinct types of contexts: ordinary, direct and indirect. The distinction is that in an ordinary context names (signs, sentences) have their ordinary denotation and sense. For example, in the sentence *Sir Walter Scott is the author of Waverley*, the name *the author of Waverley* denotes a certain man and has a particular sense, a manner of presentation. In a direct context names denote a sequence of words. For example, in the sentence *King George wrote 'Sir Walter Scott is the author of Waverley'*, the name *the author of Waverley* denotes a certain sequence of words. This is known as the use-mention distinction. Quotation marks (or *italic* letters) are commonly used to introduce a direct context. In an indirect context, however, a name denotes the sense of that same name in an ordinary context. For example in the sentence *King George wished to know whether Sir Walter Scott is the author of Waverley*, the name *the author of Waverley* denotes not its ordinary denotation, a certain man, but its ordinary sense, the manner in which the name denotes. Certain words can typically be used to generate an indirect context. For example, the words *search*, *think*, *seek*, *know*, *want*, *believe*, *etc.* introduce an indirect context.

This distinction between sense and denotation together with various types of contexts are then used by Frege (1892) to suitably limit the application of the substitutivity of equals for equals. Russell (1905) uses the following example to illustrate why it is desirable to limit the substitutivity of equals for equals in certain instances. The two sentences:

King George wished to know whether
Sir Walter Scott is the author of Waverley (1.1)

and

Sir Walter Scott is the author of Waverley (1.2)

when formalized in first-order logic and by using axioms of equality, may give rise to the conclusion that:

King George wished to know whether Sir Walter Scott is Sir Walter Scott (1.3)

The latter conclusion, though not contradictory, is highly counter-intuitive. As Russell (1905) put it: an interest in the law of identity can hardly be attributed to the first gentleman of Europe. Church (1983) referred to such conclusions as the *paradox of the name relation*. Carnap (1956) used the word *antinomy*, but the word *paradox* is preferable since no apparent contradiction occurs in the absence of any further assumptions. As pointed out by Frege (1892), the paradox is about the name relation since the fundamental problem is to explain the difference in propositions expressed by $a = a$ and $a = b$, even when $a = b$ is true.

1.2 Reasons for Eliminating the Paradox

There are several reasons for eliminating the paradox of the name relation from formal languages. First, the presence of the paradox prevents us from adopting Leibniz's definition of identity:

$$a = b \text{ stands for } F(a) \supset_F F(b) \tag{1.4}$$

This definition is to be read as: a is equal to b if for all F, $F(a)$ implies $F(b)$. The subscript F on the implication sign is to be understood as a universal quantification of F over the scope of the implication sign. If the paradox can be constructed in the language then there exist two expressions a and b such that $a = b$ and yet $\sim [F(a) \supset F(b)]$, which is in contradiction with the above definition. Second, the presence of the paradox prevents us from adopting Russell's extensionality principle for predicates:

$$P(a) =_a Q(a) \supset P = Q \tag{1.5}$$

This definition is to be read as: if P and Q are two predicates such that for all a, $P(a) = Q(a)$, then P is equal to Q. In short, two predicates are considered to be equal if they have the same truth value for all possible arguments.

In the absence of this extensionality principle, Russell's (1905) theory of contextual descriptions can be used to resolve the paradox. According to this theory most names would be considered descriptions of the form: *the x such that* $\cdots$. Descriptions do not mean anything but are merely incomplete symbols. For example, *the author of Waverley* does not mean anything but is merely an incomplete symbol. A symbol is called incomplete if it has no meaning of its own. It can have meaning only when combined with other symbols. Whitehead and Russell (1910) reason as follows. *The author of Waverley* cannot mean Scott, for then the proposition *Scott is the author of Waverley* would mean *Scott is Scott,* and that is an entirely trivial proposition. On the other hand, *the author of Waverley* cannot mean any other person, for then the proposition *Scott is the author of Waverley* would be false, and this is well known not to be the case. Consequently, *the author of Waverley* means nothing; it is an incomplete symbol.

This proof, of course, has a philosophical character and does not seem beyond doubt. So long as *the author of Waverley* has only one denotation this argument seems to be valid. The problem as pointed out by Frege (1892), however, is that *the author of Waverley* seems to have a different denotation depending on the context in which it appears. In an ordinary context it denotes a man, but in an indirect context it denotes the sense of the name *the author of Waverley*. Nevertheless, descriptions of this form may appear in any formula, and can be systematically expanded to obtain a well-formed formula. The paradox is resolved since no well-formed part of the resulting well-formed formulas can be directly identified with the original descriptions, thus making substitution of one description for another impossible. The exact method for expansion of descriptions is explained by Whitehead and Russell in Principia Mathematica and also by Church (1956). The use and expansion of descriptions is also explained in detail and applied to an example in Appendix A.

In the presence of the above extensionality principle, however, Church (1983) shows that substitution of two equal predicates in the well-formed formulas can be made, thus allowing restoration of the paradox. The details of how the paradox can be restored in the presence of the above principle is also explained in Appendix A below. Russell's (1905) resolution of the paradox of the name relation fails then if extensionality is assumed.

There are also practical reasons for eliminating the paradox of the name relation from some formal languages used in the practice of computer programming. These paradoxical conclusions can be generated at an enormous rate by a machine in the course of carrying out proofs of some theorems. Moreover, these paradoxical conclusions are (or seem to be) of no real value in the process of automatic theorem proving. No benefit is obtained by their presence in the language and depending on the set of axioms can be the source inconsistencies, as demonstrated in the previous section. They only increase the number of possible conclusions that can be derived from a set of given facts (set of clauses) making it harder for the computer to find other valid and useful conclusions. Additionally, it is important to understand the issues that are involved in implementation of a language with proper facilities for handling indirect context. Implementation of such a language will clearly illustrate not only the distinction between sense and denotation but also the distinction between functional expressions and evaluation of functional expressions.

1.3 Previous Attempts at Solution

Ajdukiewicz (1960) presents in abstract a solution to the paradox. This solution is based on the idea that sentences containing words which introduce an indirect context are ambiguous. The disambiguation process is the key to the resolution of the paradox; Church (1983) formalized this solution to the paradox. The term *proposition surrogates* is first introduced in the process of this formalization. Proposition surrogates play the role of the sense of a name; they stand in for the proposition expressed by a name (sign, sentences). In the formalized language, a proposition surrogate can be used as an agents' object of belief, knowledge, etc. The paradox is apparently resolved because different meanings of the same sentence give rise to different proposition surrogates.

Proposition surrogates make a clear distinction between the form (manner) of application of a function to its argument and the resulting value. This distinction prevents transformation of a proposition surrogate corresponding to one meaning of a sentence, to another proposition surrogate corresponding to a different meaning of the same sentence. Thus, the distinction between the form of a function applied to its argument and the corresponding value provides the key to the resolution of the paradox.

As pointed out in the concluding remarks of Church (1983), proposition surrogates fail to resolve the paradox of the name relation when applied to sentences that contain names which do not have a functional form or primitive constants. This is due to the fact that Ajdukiewicz (1960) does not account for the distinction between sense and denotation for primitive constants. Two solutions are proposed by Church (1983). One solution is to constrain the primitive constants of the formalized language so that no two are concurrent. Two primitive constants are said to be concurrent if they have the same denotation, but different senses. There is, however, no algorithm which guarantees such a condition will be true of a list of primitive constants, as there may not be enough information to determine whether two primitive constants are concurrent. For example, before Pythagoras discovery in the year 475BC, it was not known that the morning star and the evening star are actually the same star (Venus) which appears at different times of the day in the sky. The second solution is to reintroduce the sense and denotation distinction for the primitive constants of the formalized language.

1.4 Proposed Soution

In this dissertation, a solution similar to the latter one is presented. In particular, extensional entities called pointers are introduced here to play the logical role of senses of primitive constants. Thus, a pointer to a primitive constant is used to play the role of the sense of that primitive constant. Accordingly, Church's (1983) algorithm for obtaining proposition surrogates corresponding to sentences is modified. The modified proposition surrogates can then be systematically added to any formal language by means of definitions; thus requiring no modification of the underlying logic. The modified proposition surrogates are then added to Church's (1940) simple type theory and the programming language Prolog. The addition of proposition surrogates to Prolog enables the language to have a proper representation for sentences that are in an indirect context without admitting the paradox of the name relation. Proposition surrogates are used as an agent's item of belief, knowledge, etc. Prolog's standard inferencing mechanisms is then used in the process of reasoning with such sentences.

It should be noted that problems regarding selection of a set of suitable axioms for knowledge, belief, search or other words that introduce an indirect context are not addressed in this dissertation. For example questions such as: Does an agent's knowledge of a proposition imply belief as well? Can an agent know all true propositions? Does an agent know the logical conclusions of his knowledge? and other similar questions are *not* the subject of this dissertation. Whatever the ultimate answer to these questions are, however, it should be possible to state them in a formal language. Accordingly, no assumptions or answers to these questions should be dictated by the formal language, as these are mostly matters of choice. Examples demonstrating the use of the modified languages for representing sentences in an indirect context are presented.

CHAPTER 2

Alternative Interpretations of Equality

The following question should be answered before a formal language that allows representation of sentences in an indirect context can be constructed. When should two sentence, S and $S1$, be considered to express the same proposition (have the same sense)? Two sentences are said to correspond to the same item of knowledge if they express the same proposition (have the same sense). The sense of a name (sentence) may be described as that which is grasped when one understands the name (sentence) or equivalently as that which two sentences in different languages must have in common to be a correct translation of each other.

The above question can be answered in different ways. Each answer provides one alternative under which a formal language can be set up. In this chapter three different alternatives are presented. Various methods for resolving the paradox are classified according to these alternatives. These alternatives, under which two names or sentences might be considered to correspond to the same belief, item of knowledge, desire to know, etc, were first identified by Church (1951) and numbered 0, 1 and 2. Resolution of the paradox under Atl(0) is an open problem which is addressed by this dissertation.

2.1 Alternative 2

The criterion that S and $S1$ have the same sense if and only if $S = S1$ is *logically valid* is called Alt(2). Logical validity is of course subject to various explications or interpretations. Under this alternative, the equality of two names or sentences implies that their senses are also the same. Frege (1892) however presents at length many examples and reasons as to why the equality between names does not imply an equality of their senses as well. Nonetheless, this criterion is one alternative that could be adopted. This alternative is in effect one that has been adopted by Carnap (1956) as what is called the intension of a designator. It has also been adopted by Church (1951), where a formalization of the logic of sense and denotation is presented. In order to prove consistency of the formalized language, Church (1973) (1974) used a version of possible-worlds model.

The possible-worlds model has also been used more recently by a number of people, including Halpern and Moses (1985), Fagin and Halpern (1985) and Konolige (1986), to give proper semantics to various versions of modal logic. It should be pointed out, however, that some of the considered languages are propositional modal logics, with multiple agents, and thus do not directly allow quantification. Some of the proposed languages also lack the ability to deal with nested indirect context (belief about belief) as well. Therefore, there is no well-formed formula that can be said to represent sentences which involve one agent's knowledge, doubt, or belief of another.

The underlying problem with the possible-worlds model for defining semantics, however, is that it commits the agents to be, what Hintikka (1975) called, *logically omniscient*. That is, the agents must know not only all the valid propositions but also all the consequences of their knowledge. The intuitive idea behind possible-worlds model is that besides the world in which we all live, there are other worlds which are at least plausible. An agent is then said to know a proposition expressed by a sentence if that sentence happens to be true in all the different worlds considered possible by the agent. For example an agent, say King George, may consider two worlds possible. In one, Sir Walter Scott has written the Waverley novels but not in the other. However, in both of these worlds it may be true that Sir Walter Scott is a poet. King George then would know that Sir Walter Scott is a poet, but he does not know whether Sir Walter Scott is the author of Waverley. According to this model, therefore, an agent must know all the propositions expressed by sentences that are valid in all the worlds considered possible by that agent.

The possible-worlds model requires agents to also know all the logical consequences of their knowledge. For example, suppose there are a total of three possible worlds and two agents, King George and Sir Walter Scott. Let us assume that in two of the worlds, $W1$ and $W2$, both King George and Sir Walter Scott consider a sentence S to be true. Sentence S can be anything, but in particular let us say it is *Sir Walter Scott is the author of Waverley.* In the third world $W3$, however, it is assumed that neither King George nor Sir Walter Scott considers S to be true. Let us further assume that King George considers as possible only worlds $W1$ and $W3$, while Sir Walter Scott considers as possible only worlds $W1$ and $W2$.

According to the possible worlds model then, King George does not know whether Sir Walter Scott is the author of Waverley since in one of the two worlds considered possible by King George, namely $W3$, sentence S is not true. It does, however, follow that King George knows whether Sir Walter Scott knows whether Sir Walter Scott is the author of Waverley since in both worlds considered possible by King George, namely $W1$ and $W3$, Sir Walter Scott knows whether S is true or not (in $W1$ Sir Walter Scott considers S to be true and in $W3$ to be false). Computer implementations of modal logics based on the possible-worlds model have proven to be unwieldy expensive on a single processor machine in terms of time and space requirements.

Another approach to implementation of a modal logic language whose semantics is based on the possible worlds model is described by Fariñas del Cerro (1986). This approach uses resolution as the deduction method. A step called *compilation* is used during which all logical consequences of modal Horn clauses are computed and asserted to be true. These newly computed facts are then used in the process of proving certain theorems.

For example, consider a statement to the effect that Pierre knows that Jean knows that q. It is computed and asserted, during the compilation step, that

Jean knows q, that Pierre knows q, and that q are all true (regardless of what q expresses). This process of compilation is, of course, justified on the ground of the language's semantics which in turn is based on the possible-worlds model. A formal language based on the possible-worlds model then, forces certain assumptions on the agents. The validity of these assumptions however, is far from being clear. As Turing (1950) put it

> *The view that machines cannot give rise to surprises is due, I believe, to a fallacy to which philosophers and mathematicians are particularly subject. This is the assumption that as soon as a fact is presented to a mind all consequences of that fact spring into the mind simultaneously with it. It is a very useful assumption under many circumstances, but one too easily forgets that it is false. A natural consequence of doing so is that one then assumes that there is no virtue in the mere working out of consequences from data and general principles.*

It is best for a formal language, then, not to dictate any assumptions. It is better to leave such decisions to the user of the language. The formal languages presented in this dissertation do not make such assumptions. It is possible to write a set of axioms explicitly stating desired assumptions.

2.2 Alternative 1

A stronger criterion for identity between senses of two names or propositions expressed by two sentences is called Alt(1). Under this alternative, if S is convertible to $S1$ according to the rules of lambda conversion, then S and $S1$ are said to have the same sense as well. The calculi of lambda-conversion is present in Church (1941), see also chapter five of this dissertation. Though similar to Alt(2), this criterion is stronger since the senses of two names is made different whenever possible. This will become clear in the light of an example in the next section. Under Alt(1), however, if an agent believes (what is expressed by) sentence S then he must also believe (what is expressed by) another sentence $S1$ which can be obtained from S by the process of lambda conversion.

Consider for example the two sentences Iaa and $(\lambda F\lambda X\lambda Y\cdot FXY)Iaa$. These two sentences are lambda convertible to each other. Under Alt(1), if someone believes (what is expressed by) the first sentence then he must also believe (what is expressed by) the second sentence. It is possible though that the person is not familiar enough with the process of lambda conversion and is in doubt as to the equivalence of the two sentences. Of course, in certain cases the process of lambda conversion may be very difficult and lengthy to carry out even by someone familiar with the rules of lambda conversion. Frege's (1892) sense-denotation solution to the paradox of the name relation has been axiomatized and proven to be consistent under this alternative by Church (1986).

2.3 Alternative 0

The strongest criterion of identity between senses of two names or propositions expressed by two sentences is called Alt(0). Under this alternative two sentences, S and $S1$, are said to express the same sense if and only if S and $S1$ differ at most by one or more alphabetic changes of a bound variable, or one or more interchanges or partial interchanges of fully synonymous notations, including interchanges or partial interchanges of synonymous primitive constants or both. This alternative seems to be that intended by Frege (1892), since an equality between two names does not imply that their senses are the same as well. The paradox of the name relation is resolved under this alternative. At the heart of Frege's solution lies the idea that names which occur in an indirect context shift their denotation to what would be their sense in an ordinary context. Likewise, a name which occurs in a doubly indirect context will denote the sense of some unspecified name of its sense. In short, an infinite array of senses is called for since otherwise the paradox would be reconstructed at level $N + 1$, if there are only N levels of senses of a name.

Consider the example about King George that was presented in the introduction. Let us interpret the *is* in 1.2, as an equality between the two denotations of the names, *Sir Walter Scott* and *the author of Waverley*, and not between their senses. These two names in 1.1, however, occur in an indirect context; that of King George's desire to know. Thus, the two names in 1.1 denote the sense of the same names in an ordinary context, as in 1.2. Replacement of the name *Sir Walter Scott* by the name *the author of Waverley* in 1.1 is not allowed on the ground that 1.2 does not express a relation between the senses of the two names. Therefore, the paradoxical conclusion is avoided. Frege's solution to the paradox of the name relation has been formalized under Alt(2) by Church (1959) and under Alt(1) by Church (1986). This solution, though very intuitive, is yet to be formalized under Alt(0).

This dissertation examines those solutions which fall under Alt(0), or could possibly be formalized under this alternative. McCarthy (1979) solution that introduced the first-order theory of individual concepts and propositions as a formal language for knowledge-representation is such a solution. This theory is intended to avoid the paradox of the name relation. McCarthy (1979) uses two sentences:

Pat knows Mike's telephone number (2.1)

Mary has the same telephone number as Mike (2.2)

to show that a normal formalization in first-order logic will give rise to the paradoxical conclusion:

Pat knows Mary's telephone number (2.3)

and that such conclusions are not possible in his first-order theories of individual concepts and proposition. McCarthy's (1979) first-order theories, however, introduced only one level of concept of a name. For this reason, an instance of the paradox of the name relation that involves two levels of indirect context can be constructed. One such instance of the paradox of the name relation is introduced in appendix B, and it is shown that the paradox can still occur in the first-order theories of individual concepts and propositions.

Ajdukiewicz's (1960) solution to the paradox of the name relation is another solution that has been formalized under Alt(1) and Alt(0) by Church (1983). This solution is based on the idea that sentences in an indirect context are ambiguous. Ajdukiewicz's solution as well as Church's formalization of this solution are explained in detail in the next section. A shortcoming with regards to this solution is then discussed. This shortcoming can be used to reconstruct the paradox of the name relation. A modification to this solution is then presented, justified and applied. It is also proven that the use of the modified proposition surrogates does not allow the reconstruction of the paradox of the name relation. The addition of proposition surrogates to two formal languages is then presented in detail. In particular, it is shown how Prolog with the use of proposition surrogates can be used to represent and solve some classical problems in knowledge representation.

CHAPTER 3

Proposition Surrogates

Ajdukiewicz's (1960) solution to the paradox of the name relation revolves around the ambiguity of sentences similar to 2.1. It is the process of disambiguating these sentences that leads to the solution. In his attempt to find an extensional solution to the paradox of the name relation, Ajdukiewicz (1960) introduced the novel idea of abstraction with respect to a constant. The notion of abstraction with respect to a variable is familiar in most formal languages, but not abstraction with respect to a constant.

While Ajdukiewicz (1960) explains the solution to the paradox informally, Church (1983) has formalized this solution. The idea of a proposition surrogate is one that has risen out of this formalization. Ajdukiewicz (1967) had also outlined an idea very similar to Church's (1983) formulation of proposition surrogates. In this section, Church's (1983) formal notation along with Ajdukiewicz's (1960) informal explanation is applied to some examples of the paradox of the name relation. It is then shown how the paradoxical conclusions are avoided.

3.1 Development of Proposition Surrogates

Consider McCarthy's example that was previously discussed. According to Ajdukiewicz (1960), 2.1 is an ambiguous sentence and has two different meanings. One leads to the paradox of the name relation, but not the other. It should be no surprise then that most people understand the meaning of 2.1 that does not lead to the paradox, yet fail to understand the meaning that does lead to the paradox. The first meaning of 2.1 is:

> Pat knows about a particular number,
>
> about the equality relation,
>
> about Mike,
>
> about the function telephone number,
>
> that the telephone number of Mike is equal to the particular number.

The second meaning of 2.1 is

> Pat knows about a particular number,
>
> about the equality relation,
>
> that the telephone number of Mike is equal to the particular number.

Ajdukiewicz (1960) informally explained the distinction between these two different meanings of 2.1. Without the use of a formal notation it may be difficult to grasp the distinction. In what follows both an informal and a formal notation is used to shed light on this distinction. It is then shown how the second meaning leads to the paradox, whereas the first meaning avoids it.

If sentence 2.1 is understood according to the second meaning then, Pat does not explicitly know about a function (telephone number) or an individual (Mike), but only about a particular number that is the result of application of a function to an argument. According to the first meaning of sentence 2.1, however, Pat does know about an individual (Mike), a function (telephone number), and a particular number which is the result of the application of the function to the argument. As Ajdukiewicz (1960) explains, the first meaning of 2.1 is commonly understood and does not lead to the paradox. It is the second meaning of 2.1 that leads to the paradoxical conclusion. Both meanings of 2.1 are now formalized according to the notation introduced by Church (1983).

The first meaning of 2.1 is formalized under Alt(1) as:

$$know(pat, < \lambda D \lambda F \lambda G \lambda B \cdot D(\lambda X \cdot F(X, G(B))), \iota, equal, telephone, mike >) \tag{3.1}$$

and under Alt(0) as:

$$\begin{aligned} know(pat, \quad & < \lambda D \lambda F \lambda G \lambda B < D, \lambda X < F, X, < G, B >>>, \\ & \iota, equal, telephone, mike >) \end{aligned} \tag{3.2}$$

where ι is the description operator not, however, contextually defined (as in the case of Russell's descriptions), and *telephone* and *equal* are function symbols, and *mike* and *pat* are primitive constants in the language, corresponding to Mike and Pat; D, F, G, B, and X are bound variables. The distinction between proposition surrogates under Alt(0) and Alt(1) is explained below.

The first argument of *know* is a primitive constant and the second argument is an ordered n-tuple. This ordered n-tuple is referred to as a *proposition surrogate* and is the object of belief by the first argument. The first member of a proposition surrogate must be of the form $\lambda X_1 \lambda X_2 \ldots \lambda X_n \cdot M$ where M has as its only free variables exactly one free occurrence of each of the variables $X_1, X_2, \ldots, X_n$. The remaining members of the proposition surrogate must be constants $C_1, C_2, \ldots, C_n$ that are of the same type as the variables $X_1, X_2, \ldots, X_n$, but not necessarily all different. The first member of proposition surrogate gives only the form. The constants are then abstracted and listed separately. The constants are primitive symbols of the formalized language but may be names of individuals or functions.

The original formula can always be obtained from the proposition surrogate by applying the first member to the rest of the members. This device is used primarily to distinguish between application of a function to its arguments and the resulting value. For example, $2 = \sqrt{4}$ is true, whereas $2 = < \lambda F \lambda X \cdot < F, X >, \surd, 4 >$ is not true. Applying the first member of $< \lambda F \lambda X \cdot < F, X >, \surd, 4 >$ to the other members will result in $< \surd, 4 >$, and another application yields $\sqrt{4}$.

The second meaning of 2.1 is formalized under Alt(1) as:

$$know(pat, < \lambda D \lambda F \lambda A \cdot D(\lambda X \cdot F(X, A)), \iota, equal, telephone(mike) >) \quad (3.3)$$

and under Alt(0) as:

$$know(pat, \quad < \lambda D \lambda F \lambda A < D, \lambda X < F, X, A >>,$$

$$\iota, equal, telephone(mike) >) \tag{3.4}$$

The distinction between the two different meanings of 2.1, explained informally by Ajdukiewicz (1960), is vividly displayed by the above formal notation. For example, the proposition surrogates in 3.3 or 3.4 of the second meaning of 2.1, do not have separate primitive constants corresponding to a function (telephone number) and an individual (Mike) among their list of constants. The proposition surrogates of the first meaning of 2.1, in 3.1 or 3.2, however, does have constants corresponding to both the individual (Mike) and the function (telephone number) named explicitly within the list of constants.

While formalization of the first meaning of 2.1 avoids the paradox, formalization of the second meaning leads to the paradox of the name relation. Sentence 2.2 is now formalized in the usual way as:

$$telephone(mike) = telephone(mary) \tag{3.5}$$

It is possible to replace *telephone*(*mike*) by *telephone*(*mary*) in 3.4, on the basis of 3.5 and the axioms of equality, thus arriving at

$$know(pat, \ < \lambda D \lambda F \lambda A < D, \lambda X < F, X, A >>,$$

$$\iota, equal, telephone(mary) >) \tag{3.6}$$

However, no such replacement in 3.2 is possible. This is on the ground that *telephone*(*mike*) is not directly identifiable in 3.2. Indeed, 3.2 is the intended and the understood meaning of sentence 2.1.

3.2 Proposition Surrogates Under Alt(1) and Alt(0)

The distinction between formalization of proposition surrogates under Alt(0) as opposed to Alt(1) can be seen from examination of the first member of the corresponding ordered n-tuples. In case of Alt(1), no distinction is made between application of functions to arguments and the resulting values. This distinction, however, is made clear under Alt(0) by repeated use of ordered n-tuples. Under Alt(1), therefore, it may happen that the same proposition surrogate corresponds to two different sentences. For example, consider the three sentences, Iaa, $(\lambda F \lambda X \lambda Y \cdot FXY)Iaa$, and $(\lambda F \lambda X \cdot FXX)Ia$. All of these sentences are equivalent to one another under the rules of lambda conversion. The corresponding proposition surrogates under Alt(1) are respectively,

$$< \lambda G \lambda A \lambda B \cdot GAB, I, a, a >$$

$$< \lambda G \lambda A \lambda B \cdot (\lambda F \lambda X \lambda Y \cdot FXY)GAB, I, a, a >$$

$$< \lambda G \lambda A \cdot (\lambda F \lambda X \cdot FXX)GA, I, a >$$

The lambda normal form corresponding to these proposition surrogates are respectively,

$$< \lambda G \lambda A \lambda B \cdot GAB, I, a, a >$$

$$< \lambda G \lambda A \lambda B \cdot GAB, I, a, a >$$

$$< \lambda G \lambda A \cdot GAA, I, a >$$

The first two sentences have the same proposition surrogates. Under Alt(1), therefore, they express the same proposition. The corresponding proposition surrogates under Alt(0), however, are all distinct, since ordered n-tuples are used to distinguish the form of the function application from the resulting values. The corresponding proposition surrogates under Alt(0) are respectively,

$$< \lambda G \lambda A \lambda B < G, A, B >, I, a, a >$$

$$< \lambda G \lambda A \lambda B < (\lambda F \lambda X \lambda Y < F, X, Y >), G, A, B >, I, a, a >$$

$$< \lambda G \lambda A < (\lambda F \lambda X < F, X, X >), G, A >, I, a >$$

3.3 Algorithm for Obtaining Proposition Surrogates

Church (1983) describes the procedure for obtaining proposition surrogates of a sentence under Alt(0). Given any sentence the corresponding proposition surrogate can be formed. This proposition surrogate can then be used as the object of knowledge. The procedure for obtaining the proposition surrogate of a sentence S, under Alt(0) is as follows:

Step 1: Let there be n occurrences of constants in S. Let us name them $C_1, C_2, \ldots, C_n$ in order from left to right. Note these constants are not necessarily distinct.

Step 2: Let $X_1, X_2, \ldots, X_n$ be n different variables such that X_i is of the same type as C_i for $i = 1, 2, \ldots, n$, and the replacement of C_i by X_i (at the one place which is the occurrence of the constant that is referred to by C_i) will not result in the capture of the variable X_i.

Step 3: Let M be the result of replacing each C_i in S by the variable X_i. The free variables in M are then $X_1, X_2, \ldots, X_n$, each with exactly one free occurrence in M.

Step 4: Produce $M\star$ from M by replacing every occurrence in M of the notation (FA), for application of a function F to an argument A, by the notation $< F, A >$ for the ordered pair of F and A.

Step 5: The proposition surrogate is then:

$$< \lambda X_1 \lambda X_2 \ldots \lambda X_n \cdot M\star, C_1, C_2, \ldots, C_n > \tag{3.7}$$

The procedure for obtaining the proposition surrogate of a sentence under Alt(1) can be obtained by eliminating step 4 in the above procedure, and using M for $M\star$.

For example, the sentence:

Sir Walter Scott is the author of Waverley (3.8)

is formalized as: $scott = author(waverley)$ or equivalently as:

$$equal(scott, author(waverley)) \tag{3.9}$$

where *equal* and *author* are function symbols corresponding to = and author of. The constants *scott* and *waverley* correspond to Sir Walter Scott and Waverley. In sentence 1.1, King George desired to know whether the proposition expressed by sentence 3.8 is true. Thus to formalize sentence 1.1, the proposition surrogate of formula 3.9 must first be obtained. According to the above procedure the proposition surrogate of 3.9, under Alt(1) is:

$$< \lambda F \lambda A \lambda G \lambda B \cdot F(A, G(B)), equal, scott, author, waverley > \tag{3.10}$$

and under Alt(0) is:

$$< \lambda F \lambda A \lambda G \lambda B < F, A, < G, B >>, equal, scott, author, waverley > \tag{3.11}$$

sentence 1.1, can then be formalized as:

$$\begin{aligned} know(george, \quad & < \lambda F \lambda A \lambda G \lambda B < F, A, < G, B >>, \\ & equal, scott, author, waverley >) \end{aligned} \tag{3.12}$$

where *know* corresponds to wished to know whether, and *george* is a primitive constant corresponding to King George. Sentence 1.2 of the introduction section, of course, is the same as sentence 3.8 and its formalization is 3.9. The paradoxical conclusion is avoided on the ground that *author*(*waverley*) is not directly identifiable in 3.11.

Note however, that in 3.11 the primitive constant *scott* can be replaced by *author*(*waverley*) thus arriving at:

$$\begin{aligned} know(george, \quad & < \lambda F \lambda A \lambda G \lambda B < F, A, < G, B >>, \\ & equal, author(waverley), author, waverley >) \end{aligned} \tag{3.13}$$

This sentence, however, does not correspond to the paradoxical conclusion:

King George wished to know whether

$$\text{the author of Waverley is the author of Waverley} \tag{3.14}$$

on the ground that *scott* and *author*(*waverley*) are indistinguishable from each other. In fact, as pointed out earlier (with the case of 2 and $\sqrt{4}$), $scott =$ $author(waverley)$ is true. The formula corresponding to the paradoxical conclusion 3.14, under Alt(0) is:

$$know(george, \ < \lambda F \lambda G \lambda A \lambda H \lambda B < F, < G, A >, < H, B >>,$$
$$equal, author, waverley, author, waverley >) \tag{3.15}$$

Let us now consider, the case of doubly indirect context. Suppose that:

$$\text{John knows King George wished to know whether}$$
$$\text{Sir Walter Scott is the author of Waverley} \tag{3.16}$$

and

$$\text{John knows Sir Walter Scott is the author of Waverley} \tag{3.17}$$

In order to formalize 3.16 the proposition surrogate corresponding to 1.1, or the corresponding formula 3.12, must first be obtained. The proposition surrogate for formula 3.12 under Alt(0) is:

$$< \lambda H \lambda X \lambda Y \lambda Z \lambda V \lambda W < H, X,$$
$$< \lambda F \lambda A \lambda G \lambda B < F, A, < G, B >>, Y, Z, V, W >>,$$
$$know, george, equal, scott, author, waverley > \tag{3.18}$$

The formula corresponding to 3.16, then, is:

$$\begin{aligned} &know(john, \\ &\qquad < \lambda H \lambda X \lambda Y \lambda Z \lambda V \lambda W < H, X, < \lambda F \lambda A \lambda G \lambda B \\ &\qquad\qquad < F, A, < G, B >>, Y, Z, V, W >>, \\ &\qquad know, george, equal, scott, author, waverley >) \end{aligned} \tag{3.19}$$

The formula corresponding to 3.17 under Alt(0) is:

$$\begin{aligned} know(john, \; & < \lambda F \lambda A \lambda G \lambda B < F, A, < G, B >>, \\ & equal, scott, author, waverley >) \end{aligned} \tag{3.20}$$

3.4 The Problem of Proposition Surrogates

The difficulty with this solution to the paradox of the name relation lies in the way primitive constants of the formalized language are handled. An equality relation between primitive constants of the formalized language can be used to reintroduce the paradox. Consider, for example, the two sentences:

Pythagoras knows whether the morning star is the evening star (3.21)

the morning star is the evening star (3.22)

Sentence 3.21 is formalized under Alt(0) as:

$$know(pythagoras, < \lambda F \lambda A \lambda B < F, A, B >, equal, mstar, estar >) \tag{3.23}$$

where *pythagoras*, *mstar*, and *estar* are primitive constants of the language corresponding to Pythagoras, the morning star, and the evening star respectively. Sentence 3.22, is formalized in the usual way as:

$$mstar = estar \tag{3.24}$$

It is possible to substitute *mstar* for *estar* in formula 3.23 on the basis of the axiom of equality and formula 3.24, thus arriving at the conclusion:

$$know(pythagoras, < \lambda F \lambda A \lambda B < F, A, B >, equal, mstar, mstar >) \tag{3.25}$$

which corresponds to the paradoxical conclusion:

Pythagoras knows whether the morning star is the morning star (3.26)

Ajdukiewicz (1967) also pointed to the same problem but immediately dismissed it by saying that if 3.21 is true and 3.26 is false, it is because they are not understood as object-language sentences 3.23 and 3.25, but as metasentences. Such an interpretation, however, poses certain problems which are described in detail in the next chapter.

CHAPTER 4

Proposition Surrogates and Pointers

An alternative analysis of sentences containing primitive constants which avoids reconstruction of the paradox is now proposed. This solution differs from Ajdukiewicz (1967) in that it does not require metalinguistic devices (or use of quotation marks) in formalizing sentences that contain simple names. It is similar to Church's (1983) suggested solution in that it does require the distinction between sense and denotation for primitive constants. *Pointers* are introduced to play the logical role of senses of primitive constants in formal languages.

In the last section it was shown that substitution of one primitive constant of a proposition surrogate with another having the same denotation can give rise to the paradox of the name relation. Church's (1983) solution to this problem is to replace each primitive constant within a proposition surrogate by a name of its sense. The problematic substitution, then, is avoided since an equality between denotations of two constants does not imply an equality of their senses. The replacement of a primitive constant within a proposition surrogate by a name of its sense is on the ground of Frege's (1892) solution. According to this solution, names which occur in an indirect context denote their ordinary sense and not their ordinary denotation. This, of course, being an irregularity of natural languages which is to be avoided by formal languages. According to the solution of Frege (1892) and Church (1983), this irregularity of natural languages can be avoided in formal languages by introducing a new name for the sense of a name and using this new name for every indirect use of the original name.

The objection often raised to this solution is that the notion of sense of a name remains unclear as far as formal languages are concerned. Senses of names are said to be intensional as opposed to extensional, though no precise definition exists to distinguish between extension and intension. The problem of what is the sense of a name, however, can be solved by allowing pointers to play the role of senses of names in formal languages. Pointers can be understood as a special kind of description not contextually defined in the sense of Russell (1905). They describe an object by making a reference to some particular place where the object is actually described. Descriptions of this type commonly occur in written text, such as: the first sentence on the first page of this chapter. An occurrence of a primitive constant, say c_1, within a proposition surrogate is then replaced by another constant, say $@c_1$. The denotation of $@c_1$ is a pointer to the primitive constant, c_1. These two constants, c_1 and $@c_1$, are of course different.

Another property of senses of names, as explained by Frege (1892), is that the sense of a name uniquely determines its denotation. This is completely analogous to the process of pointer dereferencing in programming languages. In programming languages a pointer can be dereferenced (traced, chased) in order to determine its denotation. For example, let c_1 and c_2 be two primitive constants and two pointers to these primitive constants be $@c_1$ and $@c_2$. Equality between primitive constants does not imply equality between the corresponding pointers. I.e., $c_1 = c_2$ does not imply that $@c_1 = @c_2$. Let * be used to denote the dereferencing operation. Then $c_1 = c_2$ does imply that $*(@c_1) = *(@c_2)$. Additionally, that the sense of a name uniquely determines its denotation is expressed by $*(@c_1) = c_1$. Note that @ is not an operator, $@c$ is a name whose denotation is a pointer to c.

The algorithm for obtaining the proposition surrogate under Alt(0) as specified by Church (1983) and presented in the last section is now modified so that every occurrence of a primitive constant, say c, in the proposition surrogate is replaced by $@c$ and every occurrence of a bound variable in the body of the lambda term corresponding to the primitive constant c, is replaced by application of the * operator to that variable. For example, 3.21 will now be formalized under Alt(0) as:

$$know(pythagoras, < \lambda F \lambda A \lambda B < *F, *A, *B >, @equal, @mstar, @estar >) \tag{4.1}$$

and although $mstar = estar$, it does not follow that $@mstar = @estar$. Thus, the paradoxical conclusion:

$$know(pythagoras, < \lambda F \lambda A \lambda B < *F, *A, *B >, @equal, @mstar, @mstar >) \tag{4.2}$$

is avoided. Additionally, the proposition surrogate in 4.1 may be reduced to

$< *(@equal), *(@mstar), *(@estar) >$ which in turn can be reduced to the formula $equal(*(@mstar), *(@estar))$ and finally to $equal(mstar, estar)$.

4.1 Pointers v.s. Metalinguistic Devices

Pointers are also different from metalinguistic devices (such as quotation or other commonly used marks) to distinguish the *use* from *mention* of a symbol. There is of course a difference between talking about (mentioning) a symbol and using a symbol in any language. The usual way of distinguishing between the two is to use quotation marks whenever a symbol is talked about (mentioned) as opposed to used. The use of quotation marks to distinguish use from mention also originated from Frege (1892). The distinction is that the pointer technique withstands Langford's (1935) translation test, see also Church (1950), while the metalinguistic devices do not. The test is to translate the sentence and its analysis into a different language, say German, and to observe whether or not they convey the same meaning to one who understands German but has no knowledge of English.

The following example, from Langford (1935), illustrates the distinction. In the sentence ' 'Brot' means bread,' we shall be talking about (mentioning) the German word and using the English synonym. This sentence can be understood by one who does not understand the word 'Brot' but not by one who does not understand the English equivalent. However, ' 'Brot' means what is meant by 'bread' ' can be understood by one who knows the meaning of neither word.

Accordingly during the process of translation a word that is being used must be translated. A word that is being talked about (mentioned) must not be translated, since subject matter is to remain unchanged under translation.

Forming a name of the primitive constant in analyzing 3.21 results in:

> Pythagoras knows,
>
> about the equality relation,
>
> about the 'morning star', and
>
> about the 'evening star',
>
> whether the 'morning star' is equal to the 'evening star'.

When translating the above sentence into German, 'morning star' and 'evening star' must be translated as 'morning star' and 'evening star' not as 'Morgenstern' and 'Abendstern' since they are mentioned and not used. Translation of 3.21 into German is:

Pythagoras weiss, ob der Morgenstern der Abendstern ist. (4.3)

While translation of the above analysis of 3.21 into German is:

> Pythagoras weiss,
>
> ueber die Gleichungsrelation,
>
> ueber 'morning star' und
>
> ueber 'evening star'
>
> ob 'morning star' dem 'evening star' entspricht.

This analysis, however, is not understandable by one who understands German but has no knowledge of English. Therefore, the German translation of 3.21, i.e. 4.3, and the translation of its metalinguistic analysis do not convey the same information.

The same objection, however, does not hold for the analysis of 3.21 by way of pointers. A special list of all the primitive constants of the formalized language is first formed. Let us call this the *dictionary.* Associated with each primitive constant of the language as it appears in the dictionary, is a list of all the other primitive constants of the language that are known to be completely synonymous with it and the translation of the primitive constant in other languages. Two of the entries in the dictionary, of course, would correspond to morning star and evening star, let us assume these are entries number 10 and 15 respectively. Sentence 3.21 is then understood as follows:

> Pythagoras knows,
>
> about the equality relation,
>
> about location number 10 in the dictionary, and
>
> about location number 15 in the dictionary
>
> whether the object denoted by the primitive constant
>
> in location number 10 of the dictionary is equal to
>
> the object denoted by the primitive constant
>
> in location number 15 of the dictionary.

Translation of the above sentence into German, then, is

> Pythagoras weiss
>
> ueber die Gleichungsrelation,
>
> ueber Positionsnummer 10 im Woerterbuch und
>
> ueber Positionsnummer 15 im Woerterbuch
>
> ob das Objekt, das von der einfachen Konstanten
>
> in Position 10 des Woerterbuchs beschrieben wird
>
> dem Objekt, das von der einfachen Konstanten
>
> in Position 15 des Woerterbuchs beschrieben wird,
>
> entspricht.

Both the German translation of 3.21, i.e. 4.3, and the translation of its analysis according to pointers, however, are understandable by only one who speaks German, therefore they convey the same information.

4.2 New Algorithm for Obtaining Proposition Surrogates

The new algorithm for obtaining a proposition surrogate of sentence under Alt(0) is as follows:

Step 1: Let there be n occurrences of constants $C_1, C_2, \ldots, C_n$ in order left-to-right in S. These constants are not necessarily distinct.

Step 2: Let $X_1, X_2, \ldots, X_n$ be n different variables such that X_i is of the same type as C_i for $i = 1, 2, \ldots, n$, and the replacement of C_i by X_i (at the one place which is the occurrence of the constant that is referred to by C_i) will not result in the capture of the variable X_i.

Step 3: Let M be the result of replacing each C_i in S by the variable X_i. The free variables in M are then $X_1, X_2, \ldots, X_n$, each with exactly one free occurrence in M.

Step 4: Produce $M1$ from M by replacing every occurrence in M of the notation (FA), for application of a function F to A, by the notation $(*(F)*(A))$. Replace constants C_i and C_j, corresponding to F and A, by $@C_i$ and $@C_j$ in the list of primitive constants.

Step 5: Produce $M2$ from $M1$ by replacing every occurrence in M of the notation (FA), for application of a function F to an argument A, by the notation $< F, A >$ for the ordered pair of F and A.

The proposition surrogate then, is:

$$< \lambda X_1 \lambda X_2 \ldots \lambda Xn \cdot M2, C_1, C_2, \ldots, C_n >$$

The procedure for obtaining the proposition surrogate of a sentence under Alt(1) can be obtained by eliminating step 5 of the above procedure.

Recall that the paradox of the name relation refers to counter intuitive conclusions that may arise if concurrent names can be substituted for one another in an indirect context. Two names are said to be concurrent if they have the same denotation but different senses, e.g., morning star and evening star. If proposition surrogates are used in formalizing sentences that are in indirect context, then substitution of concurrent names within proposition surrogates is not possible.

Theorem: The new algorithm to generate proposition surrogates under Alt(0) eliminates reconstruction of the paradox of the name relation.

Proof: The construction of the paradox of the name relation requires the use of an identity between the denotation of names (either simple or functional) and substitution of one name for another when one of the names is in an indirect context. The paradoxical conclusions can also arise if a sentences that is in an indirect context can be reduced to another, according to the rules of lambda conversion. Therefore, this theorem can be proved by a separate analysis of each case where the rules of lambda conversion or the substitution of equals for equals can take place.

Case 1: Consider two wffs, **A** and **B**, and their corresponding proposition surrgates **PA** and **PB**. Let us also assume that **A** is lambda convertible to **B**. It must be shown, however, that **PA** is not lambda convertible to **PB**. The general form of a proposition surrogate, **PA**, is

$$< \lambda X_1 \lambda X_2 \ldots \lambda Xn \cdot M, \ldots >$$

This of course follows by induction from the above algorithm. Under Alt(0), unlike Alt(1), M is an ordered n-tuple. Thus, the rules of lambda conversion can not transform the proposition surrogate **A** into another proposition surrogate of the form **PB**.

Case 2: A wff, **A**, contains the constant, C_1. The constants are primitive symbols of the formalized language but may be names of individuals or functions. Let **B** be the proposition surrogate under Alt(0) of **A**, obtained according to the algorithm. Let us assume, then, that $C_1 = C_2$. It is shown that no substitution of C_2 for C_1 in **B** can take place. According to the algorithm, **B** is of the general form:

$$< \lambda X_1 \lambda X_2 \ldots \lambda Xn \cdot M, \ldots, @C_1, \ldots >$$

where $@C_1$ is a pointer to C_1, respectively. The assumption $C_1 = C_2$ is about the denotations of the names C_1, C_2 and not their pointers, $@C_1$ and $@C_2$. Thus, the equality $@C_1 = @C_2$ is not implied by $C_1 = C_2$. Additionally, the names C_1 and C_2 do not directly occur in **B**, therefore, no substitution of C_2 for C_1 can take place.

Case 3: A wff, **A**, contains a primitive constant, C_1. Let **B** be the proposition surrogate under Alt(0) of **A**, obtained according to the algorithm. Now, let us assume that the equality $C_1 = F(X)$ also holds. It is shown that no substitution of $F(X)$ for C_1 in **B** can take place. According to the algorithm, **B** is of the form:

$$< \lambda X_1 \lambda X_2 \ldots \lambda Xn \cdot M, \ldots, @C_1, \ldots >$$

where $@C_1$ is a name of a pointer to C_1. The name C_1 does not directly occur in **B**, therefore, the equality, $C_1 = F(X)$ can not imply a substitution of $F(X)$ for C_1 in **B**.

Case 4: A wff, **A**, contains application of a function to an argument of the form, $F(X)$. Let **B** be the proposition surrogate under Alt(0) of **A**, obtained according to the algorithm. Now, let us assume that $C_1 = F(X)$. It is shown that no substitution of C_1 for $F(X)$ in **B** can take place. According to the algorithm, **B** is of the form:

$$< \lambda X_1 \lambda X_2 \ldots \lambda Xn \cdot M, \ldots, @F, @X, \ldots >$$

The function application $F(X)$ does not directly occur in **B**. Therefore, the equality, $C_1 = F(X)$ can not imply a substitution of C_1 for $F(X)$ in **B**.

The above case analysis shows that an equality between the denotation of names (simple for functional) can not lead to a substitution of one name for another within a proposition surrogate. Therefore, it is not possible to reconstruct the paradox of the name relation.

An advantage of this method for resolving the paradox of the name relation is that it can be added to any formal language without modifying the underlying axioms. All that is required is to provide some kind of definition for the ordered n-tuple. Proposition surrogates can then be formed according to the above algorithm. The language's rules of inference, of course, remain unchanged. In the next two sections it is shown how proposition surrogates can be added to two formal languages.

CHAPTER 5

Formal Language

The addition of proposition surrogates to a formal language is presented in this section. The addition of proposition surrogates neither requires a change to the underlying axioms nor the rules of inference of formal languages. The proposition surrogate corresponding to a wff of a formal language can be obtained according to the algorithm presented in the last section.

When writing proposition surrogates, use is made of ordered n-tuples. The formal language, then, must be extended such that ordered n-tuples are well defined. Alternatively some other data structure, similar to ordered n-tuples, that is already present in the language can be used in place for ordered n-tuples. The formal language presented in this section, for example, has the notation for ordered n-tuples defined by means of abbreviations. When adding proposition surrogates to Prolog, however, the list data structure which is already present in the language is used in place of ordered n-tuples.

A formal language, similar to that introduced in the simple theory of types by Church (1940), is used to demonstrate how proposition surrogates can be introduced and used. For simplicity the formal language presented here will not include type symbols; all of the variables are of the same type and range over the same domain of individuals. In such a language the proposition that for any wff **A** in which **v** occurs free there is a set **u** consisting of all those elements **v** for which **A** is true, is expressed by the wff $\exists \mathbf{u} \forall \mathbf{v}[(\mathbf{v} \in \mathbf{u}) = \mathbf{A}]$. The use of equality here is extensional, i.e., if the wff to the left of the equality sign is true then the wff to its right must be true and conversely. In particular let **A** be the wff $\neg(v \in v)$, then $\exists u \forall v[(v \in u) = \neg(v \in v)]$ expresses that there is a set whose members are all those elements which are not members of themselves. Choosing such a set u, it is concluded that $(u \in u) = \neg(u \in u)$ which is a contradiction. This is called Russell's paradox, but according to the terminology adopted here it is better to call it Russell's antinomy since it gives rise to a contradiction and not just a counterintuitive result. One method for resolving this problem is the introduction of type symbols as in Whitehead and Russell (1910). The exclusion of type symbols then is a radical departure since it is well known that the resulting language is inconsistent.

Of course even a language with type symbols is not free from antinomies. Such a formal language becomes inconsistent once certain sentences are formalized. For example, formalization of sentences like Epimenides assertion that all propositions asserted by Epimenides are false or Myhill's favored proposition that all propositions favored by Myhill are false or Church's dreaded proposition that all propositions dreaded by Church are false, lead to contradictory conclusions. These are all various instances of the so called Myhill's antinomies and their removal requires introduction of some form of ramified type theory such as Whitehead and Russell's (1910), Tarski's (1933), or Church's (1956); see also Church's (1977) comparison of Russell's resolution of the semantical antinomies with that of Tarski. In order to simplify the formalized language, both simple and ramified type symbols are excluded from this presentation. The soundness and completeness of the formal language with the type symbols included, however, is shown by Henkin (1950) and also presented in detail by Andrews (1986). The syntax and semantics of the formalized language is as follows.

5.1 Primitive Symbols

1. An alphabet of primitive constants.
2. An infinite alphabet of variables.
3. The abstraction operator λ.
4. The connective () for application of function to argument.
5. The primitive constant C denoting material implication.
6. The primitive constant $\neg$ denoting negation.
7. The primitive constant Π denoting universal quantification.
8. The primitive constant ι denoting the description function.

The abstraction operator λ and the parentheses are called *improper symbols* of the language. The rest of the symbols are the *proper symbols* of the language. Material implication, negation, universal quantification, and description are taken as primitive constants. They may appear among the list of primitive constants when forming proposition surrogates of wffs. The constants of the language are primitive symbols of the formalized language and may be names of individuals or functions.

5.2 Formation Rules

1. A variable or a primitive constant standing alone is a wff.

2. If **F** and **A** are wffs then (**FA**) is a wff.

3. If **M** is a wff and **x** is a variable then (λ**xM**) is a wff.

Bold letters are used as syntactical variables. Bold capitals are used for wffs and bold lower case letters are used for variables.

5.3 Rules of Inference

1. Any part **M** of a formula can be replaced by the result of substituting **y** for **x** throughout **M**, provided that **x** is not a free variable of **M** and **y** does not occur in **M**.

2. Any part ((λ**xM**)**N**) of a formula can be replaced by the result of substituting **N** for **x** throughout **M**, provided that the bound variables of **M** are disjoint both from **x** and from the free variables of **N**. Note that **M** can always be rewritten such that **x** is not a bound variable of **M**.

3. Where **A** is the result of substituting N for **x** throughout M, any part **A** of a formula can be replaced by $((\lambda \mathbf{x}\mathbf{M})\mathbf{N})$, provided that the bound variables of **M** are distinct both from **x** and from the free variables of **N**.

4. From $\Pi\mathbf{F}$ infer **FA**, if **A** is without free variables.

5. From $C\mathbf{AB}$ as major premise and **A** as minor premise infer **B**.

The following is a detailed explanation of how to do proper substitution of M for all free occurrences of **x** in **F**, written $[\mathbf{M}/\mathbf{x}]\mathbf{F}$, with any necessary name changes. The set of *free variables* of a wff **F**, $FV(\mathbf{F})$, and *bound variables* of a wff **F**, $BV(\mathbf{F})$, are defined inductively on the structure of **F** as follows. Let **x** be a syntactical variable for the alphabet of variables and **d** be a syntactical variable for the alphabet of primitive constants.

1. $FV(\mathbf{x})$ is $\{\mathbf{x}\}$

2. $FV(\mathbf{d})$ is $\{\}$

3. $FV((\mathbf{MN}))$ is $FV(\mathbf{M}) \cup FV(\mathbf{N})$

4. $FV((\lambda \mathbf{xM}))$ is $FV(\mathbf{M}) - \{\mathbf{x}\}$

1. $BV(\mathbf{x})$ is $\{\}$

2. $BV(\mathbf{d})$ is $\{\}$

3. $BV((\mathbf{MN}))$ is $BV(\mathbf{M}) \cup BV(\mathbf{N})$

4. $BV((\lambda xM))$ is $BV(M) \cup \{x\}$

1. $[M/x]x$ is M

2. $[M/x]y$ is y *if* $y \not\equiv x$

3. $[M/x]d$ is d

4. $[M/x](AB)$ is $([M/x]A\ [M/x]B)$

5. $[M/x](\lambda xN)$ is λxN. Since x is not a free variable no substitution can take place.

6. $[M/x](\lambda yN)$ is $\lambda y[M/x]N$ if $x \notin FV(N)$ *or* $y \notin FV(M)$. That is, either no substitution of x or no capture by y is possible.

7. $[M/x](\lambda yN)$ is $\lambda z[M/x]([z/y]N)$ if $x \in FV(N)$ *and* $y \in FV(M)$, where z is a variable such that $z \notin FV(M) \cup FV(N)$. This is needed to avoid capture of bound variables.

5.4 Abbreviations

1. $A \supset B$ for CAB

2. $A = B$ for $\Pi\lambda f[fA \supset fB]$

3. $(x)A$ for $\Pi(\lambda xA)$

4. $(\exists x)A$ for $\neg(x)\neg A$

5. $< \mathbf{A}, \mathbf{B} >$ for $\lambda f(f\mathbf{AB})$

6. $< \mathbf{A}, \mathbf{A1}, \cdots, \mathbf{An} >$ for $< \mathbf{A}, < \mathbf{A1}, < \cdots \mathbf{An} >>>$

7. $\mathbf{P}(\mathbf{A}, \mathbf{A1}, \cdots, \mathbf{An})$ for $(\mathbf{PAA1} \cdots \mathbf{An})$

8. $\mathbf{A} \vee \mathbf{B}$ for $\neg\mathbf{A} \supset \mathbf{B}$

9. $\mathbf{A} \wedge \mathbf{B}$ for $\neg(\neg\mathbf{A} \vee \neg\mathbf{B})$

10. T for $(a) \cdot a \supset a$

11. F for $(a)a$

12. $\boxed{\mathbf{P}}$ for the proposition surrogate of **P** under Alt(0) obtained according to the modified algorithm presented at the end of the last section.

Abbreviations are introduced primarily due to shortness of human life and patience. The above abbreviations may help to shorten wffs and to possibly ease readability. In addition to the above abbreviations for writing wffs, the following conventions will be used. Pairs of parentheses are omitted under the convention of association to the left. A dot in a wff is an abbreviation for a pair of brackets starting at the dot and ending at the maximum distance which is consistent with the whole expression being well-formed.

The notation for ordered n-tuples is introduced by the two abbreviations $< \mathbf{A}, \mathbf{B} >$ and $< \mathbf{A}, \mathbf{A1}, \cdots, \mathbf{An} >$. These are of course similar to list data structures commonly used in programming languages. For example, $< \mathbf{A}, \mathbf{B} >$

is a function, $\lambda f(f\mathbf{AB})$, whose value with argument $(\lambda x\lambda y\cdot\mathbf{D}=x)$ is the truth-value T if $\mathbf{D}=\mathbf{A}$ and is otherwise the truth-value F, and whose value with argument $(\lambda x\lambda y\cdot\mathbf{E}=y)$ is the truth-value T if $\mathbf{E}=\mathbf{B}$ and is otherwise the truth-value F. So every element of an ordered pair can be accessed, just as any element of a list data structure is accessible, through two elementary functions whose values are the head and tail of the list.

5.5 Axioms

1. $(f)\cdot(x)(y)fxy\supset(y)(x)fxy$

2. $(f)\cdot\Pi f\supset(g)(x)f(gx)$

3. $(f)(g)\cdot(x)fx\supset gx\supset\cdot\Pi f\supset\Pi g$

4. $(f)\cdot(x)(y)fxy\supset(x)fxx$

5. $(p)\cdot p\supset(x)p$

6. $(f)(x)\cdot\Pi f\supset fx$

7. $(p)(q)(r)(s)\cdot p\supset q\supset r\supset\cdot r\supset p\supset\cdot s\supset p$

8. $(p)(q)\cdot\neg p\supset\neg q\supset\cdot q\supset p$

9. $(p)(q)\cdot p\supset q\supset\cdot q\supset p\supset\cdot p=q$

10. $(f)(g)\cdot(x)[fx=gx]\supset\cdot f=g$

11. $(f)(x) \cdot fx \supset f(\iota f)$

For a detailed statement of usage refer to Church (1940). Axioms 1 through 8, together with the five rules of inference, are sufficient for propositional calculus and the laws of quantifiers. That the formal theorems of propositional calculus are derivable from these axioms and the rules of inference was first shown by Hilbert and Ackermann (1926). Axiom 7 gives the laws of propositional calculus that involve implications only. This axiom is taken from Lukasiewicz (1947) who proved not only its sufficiency but also that it is the shortest such axiom, by examining all shorter axioms. The axioms of extensionality are 9 and 10 and 11 is the axiom of choice. These axioms were assumed by Church (1940) since they are necessary in obtaining recursive arithmetic.

The following are some theorems which are the consequences of the formal axioms.

1. $(x)fx \supset fy$

2. $fy \supset (\exists x)fx$

3. $(x)[p \supset fx] \supset \cdot p \supset (x)fx$

4. $(x)[fx \supset p] \supset \cdot (\exists x)fx \supset p$

5. $x = x$

6. $x = y \supset \cdot fx \supset fy$

7. $x = y \supset fx = fy$

8. $x = y \supset y = x$

9. $x = y \supset \cdot y = z \supset x = z$

10. $f = \lambda x(fx)$

These and a number of other related theorems were proved by Church (1940) in the process of developing the theory of recursive arithmetic. The questions regarding soundness and completeness of this language (with the type symbols included) can be answered as follows. First, a *valid* formula of the second order logic refers to a wff that has the truth-value true for every assignment of values, from an arbitrary domain of elements, to the variables and assignment of sets of ordered n-tuples to the functional variables. A language is then said to be *complete* if every valid formula of the language can ultimately be derived from the axioms, according to the inference rules of the language. Note that a completeness proof need not necessarily be constructive, i.e., provide an exact method for obtaining every valid formula of the language. A language is called *sound* if the converse is true, i.e., if only valid formulas are derivable from the axioms and the rules of inference.

According to this definition of completeness, first-order logic was proved complete by Gödel (1930). For second and higher order logics it was proved by Gödel (1931) that regardless of the choice of the underlying axioms, the language will contain wffs that are valid but not derivable from the axioms according to the rules of inference. Henkin (1950) showed, however, that a second order language with simple type theory, for example that of Church (1940), can be proved complete. In fact Henkin (1950) presents such a proof for an ω-order language by modifying the class of (standard) models which provide interpretations for the language. It is proved in particular, that for any consistent closed wff, **A**, there is a (general) model with respect to which **A** is satisfiable.

Proposition surrogates, of course, can be added to any formal language that provides some form of a notation for ordered n-tuples without changing the soundness or completeness of the language. For example, proposition surrogates are added to the language Prolog. The present purpose, however, is to give examples of how sentences in an indirect context can be represented and manipulated in this language. In the next section a number of classical examples are presented and formalized. It is then shown how the rules of inference are used in proofs of theorems that involve such sentences.

CHAPTER 6

Examples

Application of the above theory to three examples are shown in this section. The first and second examples are taken from Russell (1905) and Church (1987) to illustrate how the differences in meaning of sentences can be shown by employing a formalized language. The third example is taken from McCarthy (1978) to illustrate how the formal language can be used for representation and reasoning about agents knowledge and their reasoning about each other's knowledge.

6.1 Touchy Yacht Owner

Russell (1905) uses the following example to illustrate how elimination of contextual descriptions with respect to different scopes can give rise to different meanings of the same sentence. The same example is used here and the two different meanings are formalized according to the formal language presented in the previous section.

> *... I have heard of a touchy owner of a yacht to whom a guest, on first seeing it, remarked, 'I thought your yacht was larger than it is'; and the owner replied, 'No, my yacht is not larger than it is'. What the guest meant was, 'The size that I thought your yacht was is*

greater than the size your yacht is'; the meaning attributed to him is, 'I thought the size of of your yacht was greater than the size of your yacht'.

Let yl be a primitive constant denoting the function that returns the length of the yacht, if its argument is of the right type. Similarly, I is a primitive constants denoting the guest, and *thought* is a primitive constant which denotes a binary predicate that is true if its second argument is the proposition that is thought by its first argument. The wff corresponding to what the guest meant is

$$(\exists x)yl(x) \wedge thought(I, \boxed{(\iota y)yl(y) \wedge y > x}) \tag{6.1}$$

and the wff corresponding to the meaning attributed to the guest by the owner is

$$thought(I, \boxed{(\iota x)yl(x) \wedge (\iota y)yl(y) \wedge y > x}) \tag{6.2}$$

6.2 The Art Collector

The second example is taken from class notes of Church (1987) and is follows.

Suppose that C is a wealthy art collector and has heard of a certain painting P, one of the few paintings by the eighteenth century masters, that is still in private hands rather than in a museum. He would like to approach the owner for permission to view the painting and to possibly persuade him to sell. C therefore asks D, an art dealer, 'Who *owns P?' D replies that he does not know but will try to find out for him. D Later returns to C with the report*

E says that, O owns the painting which you are seeking (6.3)

where E is an art expert known to both C and D.

There are two rather different cases in which D's report 6.3 to C might be considered truthful. In the first, D has directly approached E, explaining C's desire to locate the painting and asking for help receiving the reply, 'O owns the painting which C is seeking.' In the second, D has attended a lecture by E in the course of which, purely by coincidence, E remarked that O owns P. The relevant difference between the two cases is that in the second, E did not know of C's desire to locate the painting and therefore could not have mentioned it.

The formalized language outlined in previous sections will be used for a more careful analysis of difference in the meaning of 6.3 in the two cases. The following propositional functions are needed. The unary propositional function *painting* such that $painting(x)$ expresses that x is a painting. The binary propositional functions *seek*, *owns*, *says* are such that $seek(x,y)$ expresses that x seeks y, $owns(x,y)$ that x owns y, and $says(x,y)$ that x said y. The first meaning of 6.3 is then understood as

$$says(E, \boxed{owns(O, (\iota x) \cdot painting(x) \wedge seeks(C, \boxed{x}))}) \tag{6.4}$$

The second meaning as

$$(\exists x) \cdot painting(x) \wedge seeks(C, \boxed{x}) \wedge says(E, \boxed{owns(O, x)}) \qquad (6.5)$$

6.3 The Wise Men

The third example is called the *The Wise Man Puzzle* and has been used to test the representational ability of formalisms for knowledge representation. The puzzle has been attributed to an unpublished note by McCarthy (1978). The source used here, however, is Konolige (1986).

A certain King wishes to determine which of his three wise men is the wisest. He arranges them in a circle so that they can see and hear each other and tells them that he will put a white or black spot on each of their foreheads, but that at least one spot will be white. In fact, all three spots are white. He then offers his favor to the one who will first tell him the color of his spot. After a while, the wisest announces that his spot is white. How does he know?

The solution to this puzzle involves reasoning about what other wise men know and do not know from observations and the king's announcements. The puzzle solved here is actually a simplified version of the above puzzle. The simplifying assumptions are that there are only two wise men, and that after some time the first wise man announces that he cannot tell the color of his spot, whereupon the second wise man says his own spot is white. The following corresponds to formulation of the puzzle.

- That A and B both have a white spot, is expressed by

$$W(A) \tag{6.6}$$

$$W(B) \tag{6.7}$$

- That each wise man knows that there is at least one white spot is expressed by

$$(x)(y)x \neq y \supset know(x, \boxed{\neg W(x) \supset W(y)}) \tag{6.8}$$

- That each wise man knows that the other wise man knows that there is at least one white spot, is expressed by

$$(x)(y)x \neq y \supset know(x, \boxed{know(y, \boxed{\neg W(x) \supset W(y)})}) \tag{6.9}$$

- That a wise man can see the other wise man's spot, is expressed by

$$(x)(y)x \neq y \supset \cdot W(x) \supset know(y, \boxed{W(x)}) \tag{6.10}$$

$$(x)(y)x \neq y \supset \cdot\neg W(x) \supset know(y, \boxed{\neg W(x)}) \tag{6.11}$$

- That a wise man knows that another wise man can observe his spot, is expressed by

$$(x)(y)x \neq y \supset know(x, \boxed{W(x) \supset know(y, \boxed{W(x)})}) \tag{6.12}$$

$$(x)(y)x \neq y \supset know(x, \boxed{\neg W(x) \supset know(y, \boxed{\neg W(x)})}) \tag{6.13}$$

- That a wise man's observation of another wise man's spot is accurate is expressed by

$$(x)(y)x \neq y \supset \cdot know(x, \boxed{W(y)}) \supset W(y) \tag{6.14}$$

$$(x)(y)x \neq y \supset \cdot know(x, \boxed{\neg W(y)}) \supset \neg W(y) \tag{6.15}$$

- That a wise man knows that another wise man's observations are accurate, is expressed by

$$(x)(y)x \neq y \supset know(x, \boxed{know(y, \boxed{W(x)}) \supset W(x)}) \tag{6.16}$$

$$(x)(y)x \neq y \supset know(x, \boxed{know(y, \boxed{\neg W(x)}) \supset \neg W(x)}) \tag{6.17}$$

- That if a wise man does not know the color of his own spot then, it must be that he knows that the color of the spot the other wise man is white, is expressed by

$$(x)(y)x \neq y \supset \cdot \neg know(x, \boxed{W(x)}) \supset know(x, \boxed{W(y)}) \tag{6.18}$$

- That a wise man knows that another wise man knows the above fact is expressed by

$$(x)(y)x \neq y \supset know(x, \boxed{\neg know(y, \boxed{W(y)}) \supset know(y, \boxed{W(x)})}) \tag{6.19}$$

- That a wise man can apply the rule of modus ponens in some particular situations, is expressed by

$$(x)(y)x \neq y \supset \cdot$$

$$know(x, \boxed{\neg know(y, \boxed{W(y)})})$$

$$\wedge$$

$$know(x, \boxed{\neg know(y, \boxed{W(y)}) \supset know(y, \boxed{W(x)})})$$

$$\supset$$

$$know(x, \boxed{know(y, \boxed{W(x)})}) \tag{6.20}$$

$$(x)(y)x \neq y \supset \cdot$$

$$know(x, \boxed{know(y, \boxed{W(x)})})$$

$$\wedge$$

$$know(x, \boxed{know(y, \boxed{W(x)}) \supset W(x)})$$

$$\supset$$

$$know(x, \boxed{W(x)}) \tag{6.21}$$

Note that these do not carry the force of an inference rule to the effect that if **x knows A $\supset$ B**, and **A** then, **x knows B**. The latter is clearly more general and if added to the language as a rule of inference has the consequence of making the agents omniscient. The above axioms, however, are specific to the particular situation in which the wise men have been placed under by the King.

- That A does not know that he has a white spot, since he made an announcement to this effect, is expressed by

$$\neg know(A, \boxed{W(A)}) \tag{6.22}$$

- That B knows that A does not know that he has a white spot, since B heard A's announcement, is expressed by

$$know(B, \boxed{\neg know(A, \boxed{W(A)})}) \tag{6.23}$$

That B knows that he has a white spot is a theorem of the above system of axioms and rules of inference. That is, $know(B, \boxed{W(B)})$, is a theorem according to the following proof.

- From 6.20 by universal instantiation and modus ponens it can be inferred that

$$\begin{gathered} know(B, \boxed{\neg know(A, \boxed{W(A)})}) \\ \wedge \\ know(B, \boxed{\neg know(A, \boxed{W(B)}) \supset know(A, \boxed{W(B)})}) \\ \supset \\ know(B, \boxed{know(A, \boxed{W(B)})}) \end{gathered} \tag{6.24}$$

which expresses that if B knows that A does not know that the color of A's spot is white and since B knows that this fact implies that A must know that the color of B's spot is white then, B would know that A knows that B has a white spot.

- From 6.19 and by universal instantiation and modus ponens it can be inferred that

$$know(B, \boxed{\neg know(A, \boxed{W(A)}) \supset know(A, \boxed{W(B)})}) \tag{6.25}$$

which expresses that B knows that if A does not know the color of his own spot then, he must know B has a white a spot.

- From 6.24, 6.25, 6.23, and by modus ponens it can be concluded that

$$know(B, \boxed{know(A, \boxed{W(B)})}) \tag{6.26}$$

which expresses that B knows that A knows that B has a white spot.

- From 6.21 by universal instantiation and modus ponens it follows that

$$\begin{gathered} know(B, \boxed{know(A, \boxed{W(B)})}) \\ \wedge \\ know(B, \boxed{know(A, \boxed{W(B)}) \supset W(B)}) \\ \supset \\ know(B, \boxed{W(B)}) \end{gathered} \tag{6.27}$$

which expresses that if B knows that A knows that B has a white spot and since B knows that this fact would imply that he has a white spot then, B would know that he has a white spot.

- From 6.16 by universal instantiation and modus ponens it follows that

$$know(B, \boxed{know(A, \boxed{W(B)}) \supset W(B)}) \tag{6.28}$$

which expresses that B knows that if A knows that B has a white spot then, B does have a white spot.

- From 6.26, 6.27, 6.28 by modus ponens it can be concluded that

$$know(B, \boxed{W(B)}) \tag{6.29}$$

Which expresses that B knows that he has a white spot.

In the next section, proposition surrogates are added to the formal language Prolog. It is then shown how each of the above axioms of the wise man puzzle can be translated into a Prolog clause, with the use of proposition surrogates. The standard inferencing mechanisms of Prolog are then used to prove the above theorem regarding B and his knowledge that he has a white spot.

CHAPTER 7

Implementation

This section presents how proposition surrogates can be used within current logic programming languages for knowledge representation. In particular, Prolog has been used, in spite of its many shortcomings in this area, as the basis for the logic programming language because of its popularity and wide accessibility. Other languages for automatic theorem proving, such as that described by Wos (1984), could have been used and perhaps should have been used since Prolog does not directly implement the axioms of equality. However, the importance and addition of these axioms to an automatic theorem proving system based on resolution has been addressed by Digricoli (1986). Implementation of proposition surrogates are discussed in the light of the wise man puzzle presented earlier.

Complete familiarity with Prolog will be assumed, for an introduction to Prolog as a programming language see Clocksin (1981) and Bratko (1986). The particular dialect of Prolog used here is called SICStus Prolog, developed by Carlsson and Widén (1988). In what follows terms starting with capital letters are Prolog's logical variables and terms starting with lower case letters are constants. Proposition surrogates are added to the language by two predicates ps(A, B) and wff(A, B). The predicate ps(A, B) is true if and only if B is the proposition surrogate under Alt(0) corresponding to wff A according to the modified algorithm for obtaining proposition surrogates as presented at the end of the section titled *Proposition Surrogates and Pointers* and is otherwise false. The predicate wff(A, B) is true if B is the wff corresponding to the proposition surrogate A and is otherwise false. Both ps(A, B) and wff(A, B) are defined in Appendix C.

A Prolog clause corresponding to each axiom of the formulation of the wise man puzzle can now be written. For example, the Prolog clause corresponding to 6.6 is

```
w(a).
```

and the clause corresponding to 6.8 is

```
know(X, P)  :-
   ps((w(Y) :- non(w(X))), P),
   diff(X, Y).
```

which is a Horn clause of the form **A** :- **B**, expressing that **A** is true if **B** is true. In this case, **B** is the conjunction of two predicates ps((w(Y) :- non(w(X))), P) and diff(X, Y). The goal predicate ps((w(Y) :- non(w(X))), P) is true iff P is the proposition surrogate under Alt(0) of w(Y) :- non(w(X)). The predicate diff(A, B) is true if A and B are different variables or constants and is otherwise false. Both predicates are defined in appendix C. Thus, the above Horn clause is a shorthand for

```
know(X, lambda([*F,     [*G,    V1], [*H,      [*J,    V2]]],
                ['@:-', ['@w', V3], ['@non', ['@w', V4]]]
               )
    ).
```

Note, however, that as a result of the call to ps/2 some goals are frozen to ensure proper evaluation of proposition surrogates in case one of the variables becomes bound. The frozen goals in this case are of the form

```
ps((w(Y) :- non(w(X))),
   lambda([*F,    [*G,   V1], [*H,     [*J,   V2]]],
          ['@:-', ['@w', V3], ['@non', ['@w', V4]]]
         )
  )
```

and

```
wff((w(Y) :- non(w(X))),
    lambda([*F,    [*G,   V1], [*H,     [*J,   V2]]],
           ['@:-', ['@w', V3], ['@non', ['@w', V4]]]
          )
   )
```

If either X or Y becomes bound as a result of further computation, then the ps goal will automatically get reevaluated which forces new binding on V1, V2, V3 or V4. Conversely, if either V2 or V4 becomes bound as a result of further computation, then the wff goal will automatically get reevaluated which forces new bindings on X or Y. The above Prolog code, in turn, corresponds to the following wff of the formalized language.

$$(x)(y)x \neq y \supset$$

$$know(x, < \lambda I \lambda F \lambda A \lambda N \lambda G \lambda B < I, lt * F, *Agt, < *N, lt * G, *B >>>,$$

$$@\supset, @W, x, @\neg, @W, y >) \quad (7.1)$$

The complete list of Prolog clauses corresponding to the formalization of the wise man puzzle is as follows:

```
1) w(a).
2) w(b).
3) know(X, P) :-
       ps(w(Y) :- non(w(X)), P),
       diff(X, Y).
4) know(X, P1) :-
       ps(w(Y) :- non(w(X)), P),
       ps(know(Y, P), P1),
       diff(X, Y).
5) know(Y, P) :-
       ps(w(X), P),
       w(X),
       diff(X, Y).
6) know(Y, P) :-
       ps(non(w(X)), P),
       non(w(X)),
       diff(X, Y).
7) know(X, P1) :-
       ps(w(X), P),
       ps(know(Y, P) :- w(X), P1),
       diff(X, Y).
8) know(X, P1) :-
       ps(non(w(X)), P),
       ps(know(Y, P) :- non(w(X)), P1),
       diff(X, Y).
9) w(Y) :-
       ps(w(Y), P),
       know(X, P),
       diff(X, Y).
10) non(w(Y)) :-
       ps(non(w(Y)), P),
       know(X, P),
       diff(X, Y).
11) know(X, P1) :-
       ps(w(X), P),
       ps(w(X) :- know(Y, P), P1),
       diff(X, Y).
12) know(X, P1) :-
       ps(non(w(X)), P),
       ps(non(w(X)) :- know(Y, P), P1),
       diff(X, Y).
13) know(X, P) :-
       ps(w(Y), P),
```

```
     ps(w(X), P1),
     non(know(X, P1)),
     diff(X, Y).
14) know(X, P2) :-
     ps(w(X), P),
     ps(w(Y), P1),
     ps(know(Y, P) :- non(know(Y, P1)), P2),
     diff(X, Y).
15) know(X, P) :-
     ps(w(X), P1),
     ps(know(Y, P1), P),
     ps(w(Y), P2),
     ps(know(Y, P1) :- non(know(Y, P2)), P3),
     know(X, P3),
     ps(non(know(Y, P2)), P4),
     know(X, P4),
     diff(X, Y).
16) know(X, P) :-
     ps(w(X), P),
     ps(w(X) :- know(Y, P), P1),
     know(X, P1),
     ps(know(Y, P), P2),
     know(X, P2).
17) non(know(a, w(a))).
18) know(b, P) :-
     ps(w(a), P1),
     ps(non(know(a, P1)), P).
```

The above set of clauses will sometimes lead to an infinite recursion since Prolog follows a depth first left to right clause evaluation order. This is because clauses (5) and (9) are mutually recursive with no boundary condition. Notice that if Prolog had a different proof search strategy, e.g. breath first, the above mutually recursive definitions would not pose a problem. This problem, however, can easily be solved by writing a top level interpreter to either change the clause evaluation order, perform depth-first search with depth bounds, or to detect circularities. Another solution is to eliminate one of the two clauses. In general, of course, it is better to write a top level interpreter since either of these clause may play an important role in proving a particular theorem.

The above set of Horn clauses were used to answer questions, in each case first the question is expressed in both English and clause form. The answers, as provided by the Prolog interpreter, are then presented.

```
% q1: Does B know that he has a white spot?
?- ps(w(b), P), know(b, P).

yes

% q2: Does B know that A has a white spot?
?- ps(w(a), P), know(b, P).

yes

% q3: Who knows that he has a white spot?
?- ps(w(X), P), know(X, P), write(X).

b
yes

% q4: Does A know that he has a white spot?
?- ps(w(a), P), know(a, P).

no

% q5: List everyone who knows that he has a white sopt.
?- ps(w(A), P), know(A, P), write(A), fail.

b
no
```

The goals q1 and q2 are self explanatory, however, q3 differs from q1 in that X is an unbound Prolog variable which gets bound to b as a result of the computation.

The goal q4 is a test to show that the set of axioms defined for this puzzle does not imply that A knows that he has a white spot, as this is known not to be the case. The goal q5 differs from q3 in that backtracking is forced to ensure that all solutions are found.

CHAPTER 8

Conclusion

This dissertation has carefully examined one of the most basic and primitive relations, the identity relation and the associated rules of substitutivity of equals for equals, which is present in formal languages including those used for programming a computer. It was shown that the common interpretation of the identity relation, i.e. a relation between only denotation of names, leads to the paradox of the name relation. Also that it is the source of inconsistencies when classical set of axioms and rules of inference are assumed by the formal language.

The thesis defended by this dissertation is that the elimination of the paradox of the name relation neither requires a modification of the classical set of axioms nor the rule of substitutivity of identity. Various solutions, under Alt(0), of the paradox of the name relation were examined and shown to be defective. One of these solutions, namely proposition surrogates, required neither modification of the axioms of the formal language nor placed any restrictions on the substitutivity of identity.

The proposition surrogate solution, however, failed to take account of the primitive constants of the formal language, names that do not have a functional form. It was shown that the proper use of pointers within the algorithm for obtaining proposition surrogates can eliminate the problem. The algorithm for obtaining proposition surrogates of a wff was accordingly modified. It was then proved that the paradox of the name relation can no longer arise if sentences in indirect context are formalized according to the new proposition surrogate algorithm.

These new proposition surrogates were added to two formal languages. It was also shown, by examples, how sentences that are in an indirect context can be formalized and how their respective rules of inferences is used in the process of reasoning with sentences that are in an indirect context.

It was shown that standard formal languages, e.g. first order logic, can be extended with proposition surrogates to deal with facts that have traditionally been expressed in modal logics. It has been argued that such facts can not adequately be expressed in standard logics; the findings and results recorded here, however, are to the contrary. In fact, the proposition surrogate approach has certain advantages over modal logics and so far no disadvantages. Proposition surrogates can be added, in a conservative manner, to standard automatic reasoning systems. This allows automatic reasoning systems to deal with facts and statements that were considered outside their domain of application.

This method of resolving the paradox and extending the formal language is of course different from the axiomatic approach. That is, ultimately it should be possible to write a set of axioms that capture the primitive ideas behind the particular method of resolving the paradox. Such a set of axioms, for example, were proposed for the Frege-Church approach to resolving the paradox and proved sound, using a version of possible worlds model, under Alt(2) by Church (1951). A similar set of axioms were proposed and proved sound, using a set-theoretic model, under Alt(1) by Church (1986). The extension of these set of axioms to Alt(0), however, seems to be a non-trivial task and is an open problem.

APPENDIX A

Contextual Descriptions

Application of Russell's contextual descriptions to resolve a paradox in the absence of the extensionality principle and restoration of the paradox in the presence of this principle, is presented here. Consider the example about Pat that was presented in the introduction. According to to Russell's (1905) solution, 2.1 should be formalized as:

$$know(Pat, (\iota x)Miketelephone(x)) \tag{A.1}$$

The expression 'Mike's telephone number' is taken to be a description and not a name. This expression denotes the x, such that x is Mike's telephone number. The symbol ι, according to Russell is an incomplete symbol of the language. When reading formulas that contain the ι symbol, it is best to replace them with a 'the'. Formula A.1 expresses the proposition that Pat knows the x, such that x is Mike's telephone number. Sentence 2.2 should be formalized as:

$$(\iota x)Miketelephone(x) = (\iota y)Marytelephone(y) \tag{A.2}$$

Formula A.2 expresses the proposition that: the x, such that x is Mike's telephone number is equal to the y, such that y is Mary's telephone number. One might be tempted to replace $(\iota x)Miketelephone(x)$ for $(\iota y)Marytelephone(y)$ in A.1

based on A.2 arriving at:

$$know(Pat, (\iota y) Marytelephone(y))$$

which is the paradoxical conclusion corresponding to 2.[illegible]

however, is not valid on the grounds that A.1 and A.2, [illegible]

formulas which correspond to the formalized language. [illegible]

hand for the well-formed formulas of the language. This [illegible]

is referred to as an incomplete symbol of the language. [illegible]

contains the ι symbol has a corresponding well-formed for[illegible]

language. The procedure for removing descriptions was s[illegible]

Principia Mathematica, see also Church (1959). According to this [illegible]

well-formed formula corresponding to A.1, is:

$$know(Pat, (\exists x).[Miketelephone(x) \equiv_x x = [illegible]$$

The subscript x under the $\equiv$ sign is used to denote unive[illegible]

notation used by Russell in Principia Mathematica. Note [illegible]

part of formula A.4, can be identified with the expression ([illegible]

This is due to the fact that descriptions are defined contextu[illegible]

formula corresponding to formula A.2, is :

$$(\exists a) \cdot [Miketelephone(x) \equiv_x x = a](\exists b) \cdot [Marytelephon[illegible]$$

The paradoxical conclusion is avoided on the grounds tha[illegible]

of formula A.4, can be identified with $(\iota x) Miketelephone$[illegible] sible to replace it with $(\iota y)Marytelephone(y)$ or its corresponding well-formed formula.

The axiom of extensionality for predicate symbols is:

$$P(x) \equiv_x Q(x) \supset P = Q \tag{A.6}$$

Addition of this axiom to the underlying logic will allow derivation of a formula that corresponds to the paradoxical conclusion 2.3. This is on the grounds that 2.2 superset $Miketelephone(x) \equiv_x Marytelephone(x)$. Based on the above axiom then $Miketelephone = Marytelephone$. The latter equality allows replacement of $Miketelephone(x)$ for $Marytelephone(x)$ in A.4, resulting in:

$$know(Pat, (\exists a) \cdot [Marytelephone(x) \equiv_x x = a]a) \tag{A.7}$$

which is the well-formed formula corresponding to the paradoxical conclusion 2.3. This example shows that the theory of descriptions cannot be adopted as a solution to the paradox of the name relation, if the axiom of extensionality for predicate symbols is admitted in the underlying logic.

APPENDIX B

McCarthy's Solution

In order to lay the groundwork for reintroducing the paradox of the name relation in the first-order theories of individual concepts and propositions, some definitions and one example from McCarthy (1979) are reproduced in this section. A new example is then presented, whose analysis in the light of the definitions given, leads however, to a paradoxical conclusion.

From the third paragraph on page 130 and conventions 1-9 of page 131 of McCarthy (1979), it is concluded that the formalized language for expressing items of knowledge has the two following linguistic domains.

Real world objects constitute a linguistic domain whose members are (or at least include) italic terms with all letters lower case. Members of this linguistic domain denote individuals in the real world. For example '*mike*' is a member of this (linguistic) domain and denotes some individual in the real world, whom (or which) we may therefore call simply *mike*. Moreover, the name '*mike*' has in addition to its quotation name also the name '*Mike*'. Thus *mike* is the denotation of the name '*Mike*', and the intention is to use the name '*Mike*' in place of the more usual quotation name "*mike*".

Concept objects constitutes a linguistic domain whose members are (or include) italic terms starting with a capital letter. Such terms are understood as denoting concepts of individuals, but the question as to what these concepts really are is left open as not being central to the theory. For example, *Mike* is the concept associated with the name 'Mike'.

Some philosophical issues concerning the nature of the members of these two domains are left open. It is clear, however, from paragraph 3 of page 130 of McCarthy (1979), that members of these two domains are not to be ordinary names, as in Frege (1892), which have both a denotation and a sense. Members of these two domains are capable only of denoting; not of connoting. Consider the example on page 129 of McCarthy (1979), which was repeated in the introduction section. McCarthy avoids the paradoxical conclusion by formalizing sentence 2.1, as:

$$know(pat, Telephone(Mike)) \tag{B.1}$$

where formula B.1 follows from formulas (1) and (10) of McCarthy (1979), with function types:

- $know$: $Realworldobjects \times Concept \rightarrow t$
- $Telephone$: $Concept \rightarrow Concept$
- $Mike \in Concept$
- $pat \in Realworldobjects$

The above notation is used for specifying the type of each function symbol and each predicate symbol. For example, the first argument of *know* is from the domain *Realworldobjects*. The second argument is from the *Concept* domain and the result is the truth-value t. The argument of the function *Telephone* is from the *Concept* domain, as is its result. Sentence 2.2 is formalized as:

$$true(Equal(Telephone(Mary), Telephone(Mike))) \tag{B.2}$$

This is formula (26) of McCarthy (1979) with function types

- $true$: $Concept \rightarrow t$
- $Equal$: $Concept \times Concept \rightarrow Concept$
- $Telephone$: $Concept \rightarrow Concept$
- $Mike \in Concept$
- $Mary \in Concept$

It is claimed that from, B.1 and B.2, it does not follow that

$$know(pat, Telephone(Mary)) \tag{B.3}$$

where B.3 follows from equations (27) and (10) of McCarthy (1979) and is the formalization of the paradoxical conclusion 2.3. Let us examine the basis of this claim. Formula B.2 seems to assert that the two concepts of Mary's telephone number and Mike's telephone number are equal to each other. If this were the case, by the axioms of equality we would be able to arrive at the paradoxical conclusion B.3, by replacing $Telephone(Mary)$ by $Telephone(Mike)$ in formula B.1. However, sentence 2.2 is not an assertion about the concepts of the telephone numbers of Mary and Mike, because it is not a sentence occurring in an indirect context. Sentence 2.2 rather, is an assertion about the two telephone numbers of Mike and Mary. Thus it is better to formalize 2.2 as

$$equal(denot(Telephone(Mary)), denot(Telephone(Mike))) \qquad \text{(B.4)}$$

where function types are as follows:

- $equal : Realworldobjects \times Realworldobjects \rightarrow t$
- $denot : Concept \rightarrow Realworldobjects$
- $Telephone : Concept \rightarrow Concept$
- $Mike \in Concept$
- $Mary \in Concept$

The well-formed formula B.3 does not follow from B.1 and B.4, because B.4 expresses an equality between the denotations of $Telephone(Mike)$ and $Telephone(Mary)$. On the other hand, B.1 is a relation between Pat and $Telephone(Mike)$. From what two formulas can formula B.3 be obtained as a conclusion? Consider the sentence

Pat knows that Mike's telephone number is the same as
Mary's telephone number (B.5)

which is formalized as

$$true(K(Pat, Equal(Telephone(Mary), Telephone(Mike)))) \tag{B.6}$$

This is formula (28) of McCarthy (1979). According to McCarthy (1979) B.1 and B.6 plus suitable axioms about knowledge will allow B.3 as a conclusion.

Consider now the following example in which formalization in the theory proceeds analogously to that of the above example. The paradoxical conclusion, however, cannot be avoided. The two sentences are:

Pat knows that Pythagoras knows whether the morning star is the evening star (B.7)

and

Pat knows that the morning star is the evening star (B.8)

the paradoxical conclusion which cannot be avoided is:

Pat knows that

Pythagoras knows whether the morning star is the morning star (B.9)

Sentence B.7 would have to be formalized in such a way that *Pat*, *Pythagoras*, *Mstar* and *Estar* are concepts associated with Pat, Pythagoras, the morning star and the evening star.

$$true(Know(Pat, Know(Pythagoras, Equal(Mstar, Estar)))) \tag{B.10}$$

where types are as follows:

- $Know : Concept \times Concept \rightarrow Concept$
- $Equal : Concept \rightarrow Concept$
- $Pat, Pythagoras, Mstar, Estar \in Concept$

Sentence B.8 would have to be formalized as

$$know(pat, Equal(Mstar, Estar)) \tag{B.11}$$

From B.10 and B.11 and the suitable axioms of knowledge as used in the example on page 129 of McCarthy (1979), we can conclude that

$$true(Know(Pat, Know(Pythagoras, Equal(Mstar, Mstar)))) \tag{B.12}$$

This is the well-formed formula corresponding to the paradoxical conclusion B.9. This conclusion can be reached on the ground that B.10 expresses a relation between $Mstar$ and $Estar$ and B.11, expresses that Pat knows that the two are equal. Thus, by the axioms of equality and the suitable axioms of knowledge, the two can be interchanged in any well-formed formula.

Sentence B.9 is not a conclusion from B.7 and B.8, according to Frege's (1892) original solution. Unlike McCarthy's first-order theories of first-order propositions and concepts, Frege allows an infinite hierarchy of concepts of names. Furthermore, every time a name occurs at level N of an indirect context, it will automatically shift its denotation to what was the sense of the same name at a level $N-1$ indirect context. Level zero of indirect context is of course the same as the ordinary context for names. In B.7 *the morning star* and *the evening start* occur in a doubly indirect context. Once in the context of Pythagoras knowing and again in the context of Pat knowing. The substitution of *the morning star* for *the evening star* in sentence B.7 is not allowed on the ground that the two names denote two different objects. In B.8 the names *the morning star* and *the evening star* occur in a singly indirect context and thus denote the sense of the same names in an ordinary context. Whereas, in B.7 the two names occur in a doubly indirect context and thus denote the sense of the same names in a singly indirect context. Note that the sense of name in a singly indirect context differs from the sense of that name in an ordinary context.

APPENDIX C

Prolog Code

```
%-------------------------------------------------------------
%  Proposition surrogates under Alt(0)
%
% The freeze mechanism is used to define proposition surrogates
% incrementaly.  Wff containing prolog variables will thus have
% the proper form.
%
% The goal ps(w(X), P) binds P to lambda([*F, A], [@w, X1])
% The goal ps(w(s(Y)), P) binds P to
% lambda([*F, [*G, A]], [@w, [@s,Y1]]).
%
% Note that the variables X, or Y do not directly occur in the
% proposition surrogate.  A two way constraint, however,
% between X, Y and X1, Y1 will ensure proper evaluation.
%-------------------------------------------------------------


:- op(250, fy,  *). /* the dereferencing operator */

%-------------------------------------------------------------
% wff(A, B) is true if B is the wff corresponding to the
% proposition surrogate A of B under Alt(0).
%-------------------------------------------------------------

wff(lambda(A, B), D) :- wff_1(A, B, C), wff_2(C, D).

%-------------------------------------------------------------
% wff_1(A, B, C) is true if C is the result of binding the
% bound variables of the lambda term A to the values in the
% list B.
%-------------------------------------------------------------

wff_1([A|As], [B|Bs], [C|Cs]) :-
   !,
   wff_11(A, B, C),
   wff_1(As, Bs, Cs).
wff_1([], [], []).

wff_11(A, B, C) :- var(A), !, wff_111(A, B, C).
wff_11(*B, B, *B) :- !.
wff_11(A, B, C) :- wff_1(A, B, C).

wff_111(A, B, C) :- var(B), !.
wff_111(A, lambda(F, G), lambda(F, G)).

%-------------------------------------------------------------
% wff_2(A, B) is true if B is the term corresponding to
% the list A such that the head of B is the first element
% of A and the arguments of B are the remaining elements of A.
%-------------------------------------------------------------

wff_2(A, B) :- wff_4(A, A1), !, construct(B, A1).
wff_2([*(P) | As], B) :-
   pointer(P),
   deref(P, A),
   atomic(A),    /* is it a predicate name? */
   !,
   wff_3(As, Cs),
   construct(B, [A | Cs]).
wff_2(*(P), A) :- pointer(P), deref(P, A).

wff_3([A | As], [C | Cs]) :- !, wff_2(A, C), wff_3(As, Cs).
wff_3([], []).

%-------------------------------------------------------------
% wff_4(A, B) is true if A is the list such that all of its
% members are either primitives or proposition surrogates and B
% is a list similar to A except that names of the form *@A are
% replaced by A.
%-------------------------------------------------------------

wff_4([A|As], [A1|A1s]) :-
   !,
   wff_41(A, A1),
   wff_4(As, A1s).
wff_4([], []).
```

```
wff_41(A, A) :- var(A), !.
wff_41(lambda(X, Y), lambda(X, Y)) :- !.
wff_41(*(A), DA) :- pointer(A), !, deref(A, DA).


%-------------------------------------------------------------
% ps(W, P) is true if P is the proposition surrogate under
% alt(0) of W
%-------------------------------------------------------------

ps(W, P) :-
   ground(W),
   !,
   ps1(W, P).
ps(W, P) :-
   ps1(W, P),
   allvars(W, X),
   freezeall(X, ps(W, P)),
   pvars(P, Y),
   freezeall(Y, wff(P, W)).

ps1(Wff, lambda(Form1, Form2)) :-
   ps_1(Wff, Form),
   /* look out for the constraints on the vars in Form2 */
   ps_2(Form, Form3, Form4),
   Form1 = Form3,
   Form2 = Form4.

%-------------------------------------------------------------
% ps_2(A, B, C) is true if B is a similar structure to A execpt
% that all primitive constants have been replaced by variables
% and C is a similar structure to A except that all primitives
% constants are replaced by pointers
%-------------------------------------------------------------

ps_2([A | B], [F1 | F2], [G1 | G2]) :-
   !,
   ps_21(A, F1, G1),
   ps_2(B, F2, G2).
ps_2([], [], []).

ps_21(*(A), V, A1) :- var(A), !. /* leave prolog vars alone */
ps_21(*(A), *(V), AP) :- atomic(A), !, pointer(A, AP).
```

```
ps_21(lambda(X, A), V, lambda(X, A)) :- !.
ps_21(A, F1, G1) :- ps_2(A, F1, G1).

%-------------------------------------------------------------
% ps_1(A, B) is true if B is a list corresponding to the
% predicate A such that the first element of B is the name of
% the predicate A and the arguments of the predicate A form the
% other members of the list B.
%-------------------------------------------------------------

ps_1(A, *(A)) :- primitive(A), !.
ps_1(A, A1) :- A =.. Bs, ps_11(Bs, B1s), ps_12(B1s, A1).

ps_12([A | B], [A1 | B1]) :- !, ps_121(A, A1), ps_12(B, B1).
ps_12([], []).

ps_121(*(A), *(A)) :- !.
/* keep the form of prop. surr. */
ps_121(lambda(A, B), lambda(A, B)) :- !.
ps_121(A, A1) :- ps_1(A, A1).

%-------------------------------------------------------------
% ps_11(A, B) is true if B is a list identical to A except that
% every primitive constant in A has been replaced another name
% of the form *A.
%-------------------------------------------------------------

ps_11([A|As], [A1|A1s]) :- !, ps_111(A, A1), ps_11(As, A1s).
ps_11([], []).

ps_111(A, *(A)) :- primitive(A), !.
ps_111(A, A).

%-------------------------------------------------------------
%
%                          UTILITIES
%
%-------------------------------------------------------------

construct(A, B) :- A =.. B.

diff(X, Y) :- X \== Y.
```

```
%------------------------------------------------------------
% like freeze(X, G), except that it works even if X
% is bound to a var
%------------------------------------------------------------

delay(X, G) :- 'SYSCALL'('$geler'(X,G)).

%------------------------------------------------------------
% A is primitive constant if it is a variable, atom or integer
%------------------------------------------------------------

primitive(A) :- var(A), !.
primitive(A) :- atomic(A).

%------------------------------------------------------------
% pointer(A, B) is true if A is an atom or
% int and B is @atom or @int
%------------------------------------------------------------

/* 64 is ASCII code for @ */
pointer(A, B) :- atomic(A), name(A, S), name(B, [64 | S]).

%------------------------------------------------------------
% pointer(P) is true if P is an atom of the form @P
%------------------------------------------------------------

pointer(P) :- atomic(P), name(P, [64 | S]).

%------------------------------------------------------------
% deref(P, A) is true if A is the dereferenced pointer P
%------------------------------------------------------------

deref(P, A) :- pointer(P), name(P, [64 | S]), name(A, S).

%------------------------------------------------------------
% ground(X) is true if X is a wff with no
% vars, except those in the prop. sur.
%------------------------------------------------------------

ground(X) :- atomic(X), !.
ground(X) :- nonvar(X), X = lambda(F, G), !.
ground(X) :- functor(X, F, N), ground1(X, 0, N).
```

```
ground1(X, N, N) :- !.
ground1(X, I0, N) :-
   I is I0 + 1,
   arg(I, X, Xi),
   ground(Xi),
   ground1(X, I, N).

%------------------------------------------------------------
% freezeall(L, P) Place a freeze on P for all the vars in L
%------------------------------------------------------------

freezeall([X|Xs], P) :- !, freeze(X, P), freezeall(Xs, P).
freezeall([], _).

%------------------------------------------------------------
% allvars(W, L) L is the list of all vars in W,
% except those occuring in prop. sur.
%------------------------------------------------------------

allvars(W, [X|Xs]) :- allVars(W, [X|Xs]-[]).

allVars(X, [X | Z]-Z) :- var(X), !.
allVars(W, X-Y) :- W =.. L, allvars_1(L, X-Y).

allvars_1([A|L], X-Y) :-
   !,
   allvars_11(A, X-X1),
   allvars_1(L, X1-Y).
allvars_1([], X-X).

allvars_11(A, [A | X]-X) :- var(A), !.
allvars_11(A, X-X) :- atomic(A), !.
allvars_11(lambda(F, G), X-X) :- !.
allvars_11(A, X-Y) :- allVars(A, X-Y).

%------------------------------------------------------------
% pvars(P, L) L is the list of all vars in the prop. sur. P
%------------------------------------------------------------

pvars(lambda(A, B), [X|Xs]) :- pvars_1(B, [X|Xs]-[]), !.
pvars(lambda(A, B), []).

pvars_1([A|B], X-Y) :- !, pvars_2(A, X-Z), pvars_1(B, Z-Y).
```

```
pvars_1([], X-X).

pvars_2(X, [X|Z]-Z) :- var(X), !.
pvars_2(X, Z-Z) :- atomic(X), !.
pvars_2(lambda(F, G), X-Y) :- !, pvars_1(G, X-Y).
pvars_2(A, X-Y) :- pvars_1(A, X-Y).

%--------------------------------------------------------------
%                          TEST GOALS
%--------------------------------------------------------------

t1 :-  ps((non(w(X)) :-  w(Y)), B), wff(B, C).

t2 :-
   ps(equal(scott, author(waverly)), A),
   ps(know(george, A), B),
   wff(B, know(george, C)),
   wff(C, D).

t3 :-  ps(equal(mstar, estar), A).

t4 :-
   ps((non(w(X)) :-  w(Y)), B),
   ps(know(Y, B), C),
   wff(C, know(Y, B)),
   wff(B, Z).

t5 :-  ps((w(Y) :- non(w(X))), B), wff(B, C).

t6 :-  ps(5, P), wff(P, A).

t7 :-  ps(w(X), P), ps(k(j, P), P1).

t8 :-  ps(w(X), P), wff(P, Q).

t9 :-  ps(w(X), P), ps(k(j, P), P1), wff(P1, Q1).

t10 :-  ps(w(X), P), wff(P, Q), ps(k(j, P), P1), wff(P1, Q1).

t11 :-
   T = w(X),
   ps(T, Q),
   X = s(Y),
   ps(T, R),
   Y = 4,
   ps(T, P),
   Q = R,
   R = P.

%-------------------------------------------------------------
% The following corresponds to the formulation of
% the wise man puzzle as presented in the Examples section
%-------------------------------------------------------------

w(a).
w(b).
know(X, P) :-
   ps((w(Y) :- non(w(X))), P),
   diff(X, Y).
know(X, P1) :-
   ps((w(Y) :- non(w(X))), P),
   ps(know(Y, P), P1),
   diff(X, Y).
know(Y, P) :-
   ps(w(X), P),
   w(X),
   diff(X, Y).
know(Y, P) :-
   ps(non(w(X)), P),
   non(w(X)),
   diff(X, Y).
know(X, P1) :-
   ps(w(X), P),
   ps((know(Y, P) :- w(X)), P1),
   diff(X, Y).
know(X, P1) :-
   ps(non(w(X)), P),
   ps((know(Y, P) :- non(w(X))), P1),
   diff(X, Y).
% Can not be added due to circularities
% w(Y) :-
%  ps(w(Y), P),
%  diff(X, Y).
% non(w(Y)) :-
%   ps(non(w(Y)), P),
%  know(X, P),
% diff(X, Y).
know(X, P1) :-
   ps(w(X), P),
   ps((w(X) :- know(Y, P)), P1),
   diff(X, Y).
know(X, P1) :-
   ps(non(w(X)), P),
   ps((non(w(X)) :- know(Y, P)), P1),
   diff(X, Y).
know(X, P) :-
   ps(w(Y), P),
   ps(w(X), P1),
   non(know(X, P1)),
   diff(X, Y).
know(X, P2) :-
   ps(w(X), P),
   ps(w(Y), P1),
   ps((know(Y, P) :- non(know(Y, P1))), P2),
   diff(X, Y).
know(X, P) :-
   ps(w(X), P1),
   ps(know(Y, P1), P),
   ps(w(Y), P2),
   ps((know(Y, P1) :- non(know(Y, P2))), P3),
   know(X, P3),
   ps(non(know(Y, P2)), P4),
   know(X, P4),
   diff(X, Y).
know(X, P) :-
   ps(w(X), P),
   ps((w(X) :- know(Y, P)), P1),
   know(X, P1),
   ps(know(Y, P), P2),
   know(X, P2).
non(know(a, w(a))).
know(b, P) :-
   ps(w(a), P1),
   ps(non(know(a, P1)), P).

%-----------------------------------------------------------
% The following corresponds to some questions that can
% automatically be answered
%-----------------------------------------------------------
```

```
% Does B know that he has a white spot? Ans is yes
q1 :- ps(w(b), P), know(b, P).

% Does B know that A has a white spot? Ans is yes
q2 :- ps(w(a), P), know(b, P).

% Who knows that he has a white spot? Ans is b
q3 :- ps(w(X), P), know(X, P), write(X).

% Does A know that he has a white spot? Ans is no
q4 :- ps(w(a), P), know(a, P).

% List everyone who knows that he has a white spot. Ans is b
q5 :- ps(w(A), P), know(A, P), write(A), fail.

% Print out everything that is known to A.
q6 :- know(a, P), wff(P, W), write(W), fail.
```

Bibliography

[Ajdu60] Kazimierz Ajdukiewicz, A method of eliminating intensional sentences and sentential formulae, pp. 17–24, in *Atti del XII Congresso Internazional di Filosofia* (1960).

[Ajdu67a] Kazimierz Ajdukiewicz, Intensional Expressions, in Jerzy Giedymin, editor, *Kazimierz Ajdukiewicz The scientific world-perspective and other essays 1931-1963*, pp. 320–347, D. Reidel Publishing Company (1967).

[Ajdu67b] Kazimierz Ajdukiewicz, Propositions as the connotation of a sentence, in Jerzy Giedymin, editor, *Kazimierz Ajdukiewicz The scientific world-perspective and other essays 1931-1963*, pp. 348–361, D. Reidel Publishing Company (1967).

[Andr63] Peter B. Andrews, A reduction of the axioms for the theory of propositional types, *Fundameta Mathematicae*, 52:345–350 (1963).

[Andr86] Peter B. Andrews, *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, Academic Press (1986).

[Brat86] Ivan Bratko, *PROLOG Programming for Artificial Intelligence*, Addison-Wesley (1986).

[Carl88] Mats Carlsson and Johan Widén, *SICStus Prolog User's Manual* (1988).

[Carn56] Rudolf Carnap, *Meaning and Necessity*, The University of Chicago Press, second edition (1956).

[Cerr86] Luis Fariñas del Cerro, MOLOG: A System That Extends PROLOG with Modal Logic, *New Generation Computing*, 4:35–50 (1986).

[Chur40] Alonzo Church, A Formulation of the simple theory of types, *The Journal of Symbolic Logic*, 5:56–68 (1940).

[Chur41] Alonzo Church, *The Calculi of Lambda-Conversion*, Princeton University Press (1941).

[Chur46] Alonzo Church, A Formulation of the Logic of Sense and Denotation, *The Journal of Symbolic Logic*, 11(1) (1946), Abstracts of Papers.

[Chur50] Alonzo Church, On Carnap's Analysis of Statements of Assertion and Belief, *Analysis*, 10(5) (1950).

[Chur51] Alonzo Church, A Formulation of the Logic of Sense and Denotation, in P. Henel, H. Kallen, and S. Langer, editors, *Structure Method and Meaning, Essays in Honor of Henry M. Sheffer*, The Liberal Arts Press (1951).

[Chur56] Alonzo Church, *Introduction to Mathematical Logic*, Princeton University Press (1956).

[Chur73] Alonzo Church, Outline of a Revised Formulation of the Logic of Sense and Denotation (Part I), *Nous*, VII(I):24–33 (1973).

[Chur74] Alonzo Church, Outline of a Revised Formulation of the Logic of Sense and Denotation (Part II), *Nous*, VIII:135–156 (1974).

[Chur76] Alonzo Church, Comparison Of Russell's Resolution Of The Semantical Antinomies With That Of Tarski, *The Journal of Symbolic Logic*, 41(4) (1976).

[Chur83] Alonzo Church, Intensionality and the Paradox of the Name Relation (March 1983), The content of this paper was presented as an invited lecture at a joint symposium of the A.P.S. and the Association for Symbolic Logic in Berkeley, California.

[Chur86] Alonzo Church, A Revised Formulation of the Logic of Sense and Denotation Under Alternative(1) (1986), Handwritten lecture notes to be published. University of California at Los Angeles.

[Chur87] Alonzo Church, An Illustration of Frege's Theory of Meaning (1987), Class notes, UCLA.

[Cloc81] W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Springer-Verlag (1981).

[Colm82] A. Colmerauer, Prolog and infinite trees, in K. Clark and S. A. Tarnlund, editors, *Logic Programming*, pp. 231–251, Academic Press (1982).

[Colm84] A. Colmerauer, Equations and inequations on finite and infinite trees, in *International Conference on Fifth Generation Computer Systems*, Tokyo, Japan (1984).

[Digr86] V. J. Digricoli and M. C. Harrison, Equality-Based Binary Resolution, *Journal of the Association for Computing Machinary*, 33(2):253–289 (1986).

[Fagi85] R. Fagin and J. Y. Halpern, Belief, Awareness, and Limited Reasoning: Preliminary Report, in *Proceedings of IJCAI-85*, Los Angeles, California (1985).

[Freg77] Gottlob Frege, Begriffsschrift, A formal language, modeled upon that of arithmatic, for pure thought, in Jean van Heijenoort, editor, *From Frege to Godel*, pp. 1–82, Harvard University Press (1977), originally published in 1879.

[Freg85] Gottlob Frege, On Sense and Meaning, in A.P.Martinich, editor, *The Philosophy of Language*, pp. 212–220, Oxford University Press (1985), originally published in 1892.

[Gill85] Marc Gillet, *Description of the P.S.C. Prolog 1.2* (1985).

[Gode77a] Kurt Godel, The completeness of the axioms of the functional calculus of logic, in Jean van Heijenoort, editor, *From Frefe to Godel*, pp. 582–591, Harvard University Press (1977), Die Vollstandigkeit der Axiom des logischen Funktionenkalkula, Monatshefte fur Mathematik und Physik, Vol. 37, pp.349-360, 1930.

[Gode77b] Kurt Godel, On Formal Undecidable Propositions of Principia Mathematica and Related Systems I, in Jean van Heijenoort, editor, *From Frefe to Godel*, pp. 592–617, Harvard University Press (1977), Uber formal unsenischeidbare Satze der Principia Mathematica und verwandter Systeme I, Monatshefte fur Mathematik und Physik, Vol. 38, pp.173-198, 1931.

[Halp85a] J. Y. Halpern and R. Fagin, A formal model of knowledge, action, and communication in distributed systems: preliminary report, pp. 224–236, in *Proceedings of the 4th ACM Symposium on the Principals of Distributed Computing* (1985).

[Halp85b] J. Y. Halpern and Y. Moses, A Guide to the Modal Logic of Knowledge and Belief: Preliminary Draft, in *Proceedings of IJCAI-85*, Los Angeles, California (1985).

[Henk49] Leon Henkin, The Completeness Of The First-Order Functional Calculus, *The Journal of Symbolic Logic*, 14(3):159–166 (1949).

[Henk50] Leon Henkin, Completeness In The Theory of Types, *The Journal of Symbolic Logic*, 15(2):81–91 (1950).

[Henk63] Leon Henkin, A theory of propositional types, *Fundameta Mathematicae*, 52:323–344 (1963), See also the Errata to this paper in Fundameta Mathematicae, Vol. 53, pp. 119, 1963.

[Hint75] J. Hintikka, Impossible possible worlds vindicated, *Journal of Philosophical Logic*, 4:475–484 (1975).

[Kono86] Kurt Konoligue, *A Deduction Model of Belief*, Morgan Kaufmann Publishers (1986).

[Krip63] S. Kripke, Semantical analysis of modal lögic, *Zeitschrift fuer Mathematische Logik und Grundlagen der Mathematik*, 9:67–97 (1963).

[Lang35] C. H. Langford, Reviews, *The Jornal of Symbolic Logic*, 2:53 (1935).

[Lloy84] J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag (1984).

[Luka70] Jan Lukasiewicz, The Shortest Axiom of the Implicational Propositional Calculus, in L. Borkowski, editor, *Jan Lukasiewicz Selected Works*, pp. 295–305, North-Holland Publications (1970), originally published in 1947.

[McCa79] John McCarthy, First Order Theories of Individual Concepts and Propositions, in B. Meltzer and D. Michie, editors, *Machine Intelligence 9*, pp. 120–147, Ellis Horwood (1979).

[Park88] Stott Parker and Muntz R. R., *A Theory of Directed Logic Programs and Streams*, Technical Report CSD-880031, Computer Science Department University of California Los Angeles (April 1988).

[Quin56] W. V. Quine, Unification of universes in set theory, *The Journal of Symbolic Logic*, 21:267–279 (1956).

[Robi65] J. A. Robinson, A machine-orinted logic based on the resolution principle, *Journal of ACM*, 12(1):23–41 (1965).

[Russ05] Bertrand Russell, On Denoting, *Mind*, 14:479–493 (1905).

[Skol23] Thoralf Skolem, The foundation of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains, in Jean van Heijenroot, editor, *From Frege to Godel*, pp. 302–333, Harvard university press (1923).

[Tars23] A. Tarski, Sur le terme primitif de la logistique, *Fundamenta Mathematicae*, 4:196–200 (1923).

[Tars56] Alfred Tarski, The Concept of Truth in Formalized Languages, in *Logic, semantics, methamathematics, Papers from 1923 to 1938, by Alfred Tarski*, pp. 152–278, Hackett Publishing Company (1956).

[Turi50] Allen M. Turing, Computing machinery and intelligence, *Mind*, 59:433–460 (1950).

[Whit13] Alfred North Whitehead and Bertrand Russell, *Principia Mathematica*, Cambridge press (1910-1913).

[Wos69] Larry A. Wos and G. A. Robinson, Paramodulation and theorem proving in 1st order theories with equality, in B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pp. 135–150, Edinburgh University Press, Edinburgh, Scotland (1969).

[Wos80] Larry A. Wos, R. A. Overbeek, and L. Henschen, Hyperparamodulation: A refinement of paramodulation, pp. 208–219, in *Proceedings of the 5th Conference on Automated Deduction*, Springer-Verlag, Les Arcs, France (July 1980).

[Wos84] Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle, *Automated Reasoning: Introduction and Applications*, Prentice-Hall Inc. (1984).