Abstract

Title of Dissertation: Step-logic: Reasoning Situated in Time
Author: Jennifer J. Elgot-Drapkin, Doctor of Philosophy, 1988
Dissertation directed by: Donald Perlis, Associate Professor, Department of Computer Science

The world in which a commonsense reasoning agent reasons, that is, the everyday world, is continually changing. These changes occur as the agent proceeds, and must be taken into account as the agent reasons. An often overlooked, but extremely important, change that occurs is simply the *passage of time* as the agent reasons.

A commonsense reasoner is frequently limited in the amount of time it has to reason. Conclusions that may be logically (or otherwise) entailed by the agent's information take time to be derived. But time spent in such derivations is concurrent with changes in the world. This limitation must be recognized by the reasoner himself; that is, the agent should be able to reason about its ongoing reasoning efforts themselves. To do this, the agent's reasoning must be ''situated'' in a temporal environment.

The problem that I address is that of defining a formalism in which the on-going process of deduction itself is part of that very same reasoning. This involves focusing on individual deductive steps, rather than the collection of all conclusions ever reached. This has led to the formulation of step (or situated) logic, an approach to reasoning in which the formalism has a kind of real-time self-reference that affects the course of deduction itself. Such a notion of logic deviates in a crucial way from traditional formal deductive mechanisms, for the proof *process* becomes part of the available information used in forming proofs.

A precise characterization of step-logic is given, with details of two particular step-logics. Two commonsense reasoning problems, the *Brother problem* and the *Three-wise-men problem*, are modelled using step-logic, providing real-time formal solutions to these commonsense reasoning problems. These solutions were then implemented on an IBM PC-AT.

It appears that step-logic is a promising formalism for modelling the fact that reasoning takes time. Contradictions can arise and be subsequently resolved within the logic itself, permitting a genuinely computational solution to certain types of default reasoning.

STEP-LOGIC:
Reasoning Situated in Time

by
Jennifer J. Elgot-Drapkin

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1988

To my dad, Calvin C. Elgot.

Acknowledgements

I would like to thank my husband, Michael S. Drapkin, for his continued love, patience, and support over the last few years. On many occasions our discussions centered on my research endeavors, forcing his problems to take a back seat. He spent many a late night discussing step-logic with me, even though he had to be in to work first thing in the morning. Even when we were living three states apart, he would spend countless hours on the telephone keeping me going (and the phone company in business). I never would have been able to complete this degree without his help.

Special thanks go to my advisor, Don Perlis. He was amazingly patient and never seemed to lose faith in me. He survived the long late hours that resulted when our ‘‘one-hour meeting’’ stretched into eight. Don has been an inspiration to me all these years. I can only hope that some day I too may be as helpful to my students as he was to me.

I would like to thank the members of my advisory committee, Don Perlis, Jack Minker, Bill Gasarch, Dana Nau, and Ken Lopez-Escobar, for their time, efforts, and criticisms. They truly helped shape this dissertation.

I would also like to thank my family and friends who were always there when I needed them---thanks for putting up with me.

Finally, I wish to express my thanks to my daughter Amy who forced me to wrap things up---I'm so glad you waited until eleven days *after* my defense to be born!

Contents

List of Figures

1 Introduction

This dissertation attempts to construct an appropriate deductive formalism for real-time commonsense reasoners (frequently referred to in this thesis as ''reasoning agents'', or more simply, ''agents''). This chapter provides an introduction to and motivation for the work. Section 1.1 describes what the problem is and what I intend to do about it. Section 1.2 sketches my major contributions to the problem. A brief synopsis of the chapters is given in Section 1.3.

1.1 Statement of the Problem

What is meant by a *real-time* commonsense reasoner? The world in which a commonsense reasoning agent reasons, that is, the everyday world, is continually changing. These changes occur as the agent proceeds, and must be taken into account as the agent reasons. An often overlooked, but extremely important, change that occurs is simply the *passage of time as the agent reasons*. By ''real-time commonsense reasoner'' I mean one that takes this passage of time into account.

A real-time commonsense reasoning agent's logic must be able to evolve and represent the evolving history at the same time. That is, since time goes on as the agent reasons, and since this phenomenon is part of that about which the agent must reason, the agent will need to take note of facts that come and go. This immediately puts us in a non-traditional setting, for we lose monotonicity: as the history evolves, some conclusions are lost. As an example, the agent may have the belief that ''It's currently 12:00.'' We do not want this to remain a belief when it is 12:15. It is thus important for the reasoning agent to be able to recognize this passage of time.

The issue of representing time within a logic has been studied intensively, e.g., by Allen [All84], McDermott [McD82], and McKenzie and Snodgrass [MS87]. However, such representations of time are not related in any obvious way to the process of actually producing theorems in that *same* logic. We might call this type of time study *static*, for it does not require non-monotonicity. What we are studying in this thesis is quite a different sort of time.

A commonsense reasoner is frequently limited in the amount of time it has to reason. Conclusions that may be logically (or otherwise) entailed by the agent's information (beliefs) take time to be derived. But time spent in such derivations is concurrent with changes in the world. Thus, for example, it is not appropriate to spend hours figuring out a plan to save Nell from an onrushing train; she will no longer need saving by then (see [McD82] and [Haa85]). Even if the only changes are within the agent, this is still important, for it may be useful to know whether a problem is nearing solution, or if one has only begun initial explorations, and so on.

This limitation must be recognized by the reasoner himself; that is, the agent should be able to reason about its ongoing reasoning efforts themselves. Its reasoning must be ''situated'' in a temporal environment (see [Suc86]). The paradigm for such an agent would seem to be that suggested by [Nil83], namely, a computer individual with a lifetime of its own. What is of interest for such an agent is not its ''ultimate'' set of conclusions, but rather its changing set of conclusions over time. In fact there will be, in general, no ultimate or limiting set of conclusions.

Already existent formalisms for commonsense reasoning do not give the reasoner the ability to recognize that its reasoning takes time. The problem then that I address is that of defining a formalism in which the on-going process of deduction itself is part of that very same reasoning. Step-logic is proposed as such a model of reasoning.

Such a notion of logic deviates in a crucial way from traditional formal deductive mechanisms, for the proof *process* will now be made part of the available information used in forming proofs. In simple terms, there will be a notion of ''now''. Just how this can be formalized and applied to Artificial Intelligence is the principal theme of this dissertation. It is this introduction of ''now'' that creates the built-in kind of non-monotonicity that we mentionned earlier. This is quite different from the usual default kind of non-monotonicity. Yet this temporal kind of non-monotonicity lends itself to a treatment of the default kind, as we will see in Chapter 6.

Commonsense reasoners must deal with a world about which incomplete information is known. Hence conclusions are frequently derived without total information. These default conclusions may later contradict inferences that are based on new information. In traditional logics, contradictions must be avoided at all costs for they cause what we call the *swamping problem*: from a contradiction, all wffs come to be theorems. This obviously is inappropriate for our commonsense reasoner. A suitable formalism must provide a method for resolving the resulting contradictions.[1] This may require retracting prior conclusions. Because of this, traditional monotonic logics are unsuitable for formalizing these agents; we are again forced into a non-monotonic setting. Although several formalisms for non-monotonic reasoning have already been proposed (see [MD80, Rei80, McC80]), they do not address the issue of *real-time* commonsense reasoning.

1.2 Accomplishments

The primary contributions of this work are the following:

- A precise characterization of step-logic, a formalism designed to take into account the fact that reasoning occurs over time, is given.

  Two distinct types of formalisms are given: the meta-theory SL^n *about* the reasoning agent, and the agent-theory SL_n itself. The agent-theory is step-like; the meta-theory is simply our assurance that we have been honest in describing what we mean by a particular agent's reasoning. The two theories together form a step-logic *pair*. The subscript n serves to distinguish different versions of the step-logics. The versions differ in the mechanisms that the agent has at its disposal: self-knowledge, time, and retraction. Self-knowledge gives the agent the capability to introspect and determine what it does and does not know. The time mechanism allows the on-going process of deduction to be part of the agent's own reasoning. Retractions are necessary because our logic is non-monotonic.

- Theorems about step-logics are stated and proven.

  The most notable theorem perhaps is that of analytic completeness. A meta-theory is *analytically complete* if for each natural number i and for each α in the agent's language, the meta-theory either can prove that α is a theorem of the agent's at step i or can prove that α is not a theorem of the agent's at step i. SL^0 is shown to be analytically complete (see Chapter 4).

- Step-logic is shown to be a formalism in which contradictions can arise and be subsequently resolved *within the formalism itself.*

  Contradictions do not need to be avoided since they do not cause the swamping problem. Indirect contradictions are allowed to persist. Direct contradictions, which are easily identified, are resolved in a simple manner.[2]

- Two step-logics (SL^0 and SL_7) are studied in detail.

  SL^0 is the meta-theory corresponding to the simplest agent-theory, SL_0. SL_0 is a mere propositional reasoner. SL^0 was studied to gain an understanding of step-logic in general. See Chapter 4. SL_7 is the most sophisticated agent-theory: it has all three of the proposed mechanisms, self-knowledge, time, and retraction. It is described in Chapter 5.

[1]Note that in general, the problem of identifying implicit contradictions is undecidable. In the formalism that we propose, only direct contradictions need to be identified.

[2]Only certain types of direct contradictions are currently resolved.

- Step-logics are applied to two commonsense reasoning situations.

  Moore's *Brother problem* is described and a formal solution is presented in Chapter 6. The *Three-wise-men problem* is formalized and a solution using step-logic is given in Chapter 7.

- Implementations for several step-logics are given.

  Prolog implementations were written for SL^0 and two different SL_7 logics: one used to solve the *Brother problem*, and one used to solve the *Three-wise-men problem*. All programs were run on a DEC VAX 11-780 as well as an IBM PC-AT. The implementations are described in Appendix D and sample runs are given in Appendix C.

We will see then that step-logic provides a promising formalism for modelling the fact that reasoning takes time. Contradictions can arise and be subsequently resolved within the logic itself. This permits a genuinely computational solution to certain types of default reasoning. Two commonsense reasoning problems, the *Brother problem* and the *Three-wise-men problem*, are modelled using step-logic.

1.3 Outline of Thesis

In this section I briefly sketch the research which this dissertation encompasses. Some of the work reported on includes work done jointly with my advisor; see [DP86b, DP86a, EDP88].

Chapter 2 provides an overview of step-logic. I describe how it is different from traditional logic, and why it is that this new logic is needed. Reasoning agents that are logically omniscient are described and contrasted with those that have limited reasoning capabilities. Step-logic is then contrasted with already existent approaches to limited reasoning.

Chapter 3 describes the proposed step-logics in detail. It is postulated that an agent requires at least three major mechanisms to be able to deal with most commonsense reasoning issues. Each of the step-logics differs in the mechanisms that the agent has at its disposal. Each step-logic which models an agent-theory can have a corresponding step-logic for the meta-theory; the two step-logics together form what is called a step-logic *pair*. This chapter also discusses the notion of completeness of the various step-logics.

$< SL_0, SL^0 >$ is the first of the step-logic pairs. It is the simplest. The agent's theory, SL_0, is merely propositional logic, but the meta-theory, SL^0, incorporates the idea that reasoning occurs in steps. Chapter 4 provides details of the operation of SL_0 and SL^0. A theorem about the completeness of SL^0 is proven. As practical evidence that, in fact, SL^0 correctly describes the operation of SL_0, an implementation is also given.

SL_7 is the most sophisticated agent-theory: it has all three proposed mechanisms. It has successfully been used to model both Moore's *Brother problem* and the *Three-wise-men problem*. Chapter 5 describes SL_7 in detail, and Chapters 6 and 7 describe how SL_7 is used to solve the *Brother problem* and the *Three-wise-men problem*, respectively. Implementations are given for each of the two problems.

Chapter 8 summarizes the research, and discusses some of the difficulties with and limitations of the particular step-logics studied. Possible areas for further research are then suggested.

Appendices A and B contain the proofs of various theorems of SL^0 and SL_7, respectively. Appendix C contains actual runs for the various implementations of the step-logics, and Appendix D contains the code for the various implementations.

2 A Time-situated View of Reasoning

This chapter provides an overview of step-logic. Step-logic is intended as a formalism for commonsense reasoners. As such, we would like it to serve as a model of an agent with limited capabilities whose reasoning is situated in a temporal environment. Logical omniscience is not an appropriate assumption for such reasoners. Section 2.1.1 contrasts logical omniscience with limited reasoning, and describes some of the work that has been done in the area of *limited* reasoning. Section 2.1.2 sketches the work that has been done in *temporal* reasoning. Section 2.2 describes the general idea behind step-logic, and explains how it is different from any other limited or temporal reasoning formalism. Section 2.3 reviews several key problems in commonsense reasoning, explaining why step-logic may be a useful tool for modelling these problems.

2.1 Background

2.1.1 Logical omniscience vs. Limited reasoning

Traditional logics are not suitable for modelling beliefs of reasoning agents with limited capabilities because they suffer from the problem of logical omniscience: if an agent has $\alpha_1, \ldots, \alpha_n$ in its belief set, and if β, a wff of the agent's language, is logically entailed by $\alpha_1, \ldots, \alpha_n$, then the agent will also believe β. As a specific example, if an omniscient agent believes α, and also believes $\alpha \to \beta$, then the agent will believe β.

As an illustration of this conventional omniscient approach, refer to Figure 2.1. Here we see that the

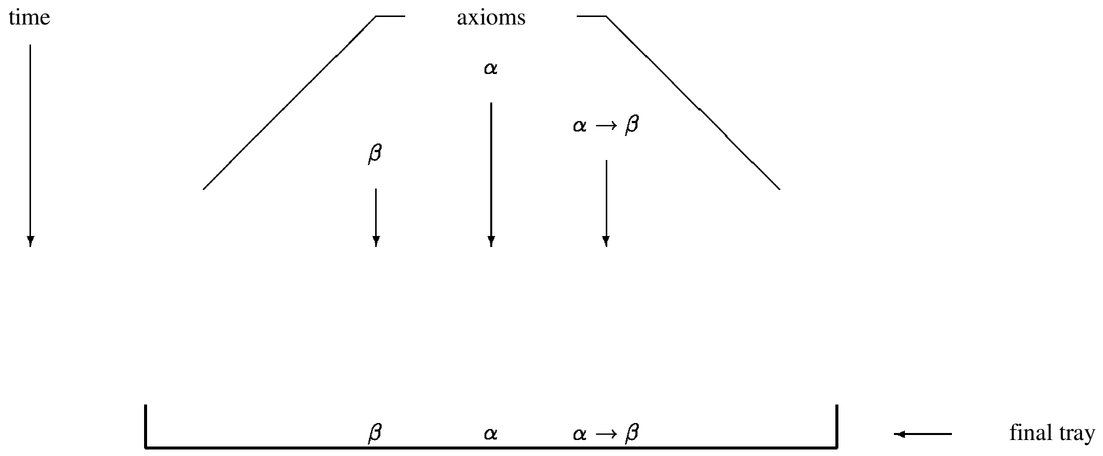

Figure 2.1: Final-tray logical studies

reasoner begins with an initial set of axioms, and the deductive mechanism generates theorems along the way. The agent concludes α, and $\alpha \to \beta$; later on, since the agent is logically omniscient, it will conclude β. One typically views the logic, however, in terms of a final tray of conclusions; this is the bar at the bottom of the figure. One is not generally aware that α, for instance, is being deduced ‘‘before’’ $\alpha \to \beta$. We can think of the theorems as ‘‘dropping down’’ and being ‘‘caught’’ by the tray. One notes that α, $\alpha \to \beta$, and β are all theorems of the logic, and is not concerned with the order in which each wff is deduced. This is the traditional *omniscient* approach.

Commonsense reasoners have limited reasoning capabilities because they must deal with a world about which they have incomplete knowledge. Hence traditional logics cannot be used as satisfactory models of these agents. This has brought about tremendous strides in the area of *limited reasoning*. We describe here several such approaches.

[Kon84] includes work based on the fact that people are not always aware of all the relevant rules. A small child, for instance, may be trying to solve the equation ‘‘$x + 2 = 5$’’. If the child does not know that, in order to solve for x, he must subtract two from both sides of the equation, he may be unable to solve it. His capabilities may be limited by the fact that he does not know all the possible rules. Konolige used this idea of *relevance incompleteness* to get an interesting solution to the *Three-wise-men problem*. In Chapter 7 we present our own solution to this problem.

[Kon84] also discusses two other types of limited reasoning. The first is *resource-limited incompleteness*, in which an agent may have the inferential capability to derive some consequence of his beliefs, yet does not have the computational resources to do so. This is a difficulty that arises, for instance, in a chess-playing program.

The final type of limited reasoning that [Kon84] mentions is that of *fundamental logical incompleteness*. It may be, for example, that an agent simply does not reason about beliefs of other agents at all. This would make a solution to the *Three-wise-men problem* impossible.

Levesque ([Lev84]) has given an intuitively plausible semantic account of *implicit* and *explicit* beliefs. An agent’s implicit beliefs include all valid formulas, his explicit beliefs, and the logical consequences of his explicit beliefs. His explicit beliefs, on the other hand, are closed under a much weaker set of conditions. An agent does not necessarily explicitly believe all valid formulas, nor does it necessarily explicitly believe β, simply because it explicitly believes α and $\alpha \rightarrow \beta$. Using the set of explicit beliefs, Levesque is able to describe an agent who is not logically omniscient.

Levesque’s logic, however, does not allow meta-reasoning about one’s own beliefs or reasoning about other agents’ beliefs. These abilities are needed in many situations, including planning and goal-directed behavior, where one may have to reason about the knowledge that one has as well as the knowledge that others may possess. Fagin and Halpern ([FH88]) extend Levesque’s notion of implicit and explicit beliefs to allow for multiple agents and beliefs of beliefs.

[FH88] propose a logic of awareness (again with explicit and implicit beliefs) which is based on the idea that one cannot have beliefs about something of which one has no knowledge. Intuitively, given a primitive proposition p, if the agent explicitly believes $p \vee \neg p$ then the agent is aware of p. As in [Lev84]’s logic, an agent’s implicit beliefs include all valid formulas and all the logical consequences of his implicit beliefs. The explicit beliefs, on the other hand, are generated by awareness of primitive propositions. As in [Lev84], the explicit beliefs do not necessarily include all valid formulas but, unlike [Lev84], *are* closed under implication.

[FH88] extend their logic of awareness to include awareness of arbitrary formulas (not just primitive propositions). In addition to the operators for implicit and explicit belief (L_i and B_i, respectively), an operator for awareness, A_i, is introduced. An agent explicitly believes a formula α if he implicitly believes α and he is aware of α (that is, $B_i\alpha \longleftrightarrow L_i\alpha \wedge A_i\alpha$). As in the previous logic, an agent does not explicitly believe all valid formulas; however, unlike the previous logic, an agent’s explicit beliefs are not necessarily closed under implication. Thus it is possible for an agent to explicitly believe p and $p \rightarrow q$ without explicitly believing q. The intuitive explanation given for this is that the agent is not aware of q.[1] It is interesting to note that L_i acts like the classical belief operator, so that, for instance, if one assumes that the agents are aware of all formulas, the logic reduces to the classical logic of belief, weak S5 (see [Che80]).

[FH88] also present a logic of local reasoning which allows agents to hold inconsistent beliefs. It is based on the fact that humans don’t focus on all issues simultaneously. Thus one can view a reasoning agent as a society of minds, each with its own set of beliefs. Unlike the previous logics that [FH88] propose, in this logic there is not necessarily only one set of states that an agent thinks possible, but rather many sets, each one corresponding to a different set of beliefs. That is, each set represents the ‘‘worlds’’ the agents

[1]This author feels that it is a little odd to say that we can be aware of $p \rightarrow q$ (which the agent must, since he explicitly believes $p \rightarrow q$) without being aware of q. The notion of awareness which is presented in this dissertation does not allow this peculiarity.

thinks are possible in a given frame of mind, when focusing on a certain set of issues. It is then possible for conclusions that are drawn in one world to be inconsistent with conclusions drawn in another.

Although these approaches all model *limited* reasoning, the process is still in terms of the standard mold of *static* reasoning. We do indeed have a restricted view of what counts as a theorem, but the logic is still final-tray-like. Although the final tray is smaller than in the conventional *omniscient* approach (it is catching less, if you will), it is still only the *final* set of consequences that are evident. In Fagin and Halpern's logic of general awareness ([FH88]), for example, $\alpha, \alpha \rightarrow \beta, \alpha \rightarrow \gamma$ and γ may all appear in the tray without β, given that the agent is unaware of β. Although the tray is catching less here, the over-simplification of a ‘‘final’’ state of reasoning is nonetheless maintained. All the conclusions are still drawn instantaneously. The effort involved in actually performing the deductions is not taken into consideration.

2.1.2 Reasoning about Time and Action

Formal reasoning about time and action is not new. A great deal of research has been devoted to this field. Perhaps the two most influential temporal formalisms are those of Allen ([All84]) and McDermott ([McD82]).

In [All84], Allen develops a logic which permits reasoning about time. Time intervals are the principal objects in the domain. Three basic entities are associated with time: properties, events, and processes. $HOLD(p, i)$, where p is a property type (e.g. red) and i is an interval of time, is used to denote the fact that property p holds for the interval i. A property is true for an interval iff it holds for every subinterval. The fact that an event e occurred over an interval i is denoted by $OCCUR(e, i)$. Finally, $OCCURRING(p, i)$ denotes the fact that process p occurred over interval i. A process is said to occur over an interval i iff it occurs over some subinterval of i. Having set up a way to handle temporal information, [All84] then proceeds to handle actions, causation, intentions, and plans.

[McD82] constructs his theory using *fact types* and *event types*. Unlike Allen, McDermott uses time *points* as primitive. $T(t, p)$ denotes the fact that fact type p holds at time t. $OCC(t_1, t_2, e)$ denotes the fact that event type e occurred over the interval $< t_1, t_2 >$. [McD82] then uses these primitives to reason about temporal information and events.

Many others have contributed formalisms for reasoning about time and action including [Haa86, HS86, MB83, MS87, Moo85, Lif87, Sho88]. In contrast to these theories, the focus of this thesis is not primarily to be able to reason *about* time, but rather to be able to reason *in* time. That is, step-logic is introduced as a time-situated view of reasoning, where the fact that the reasoning process itself takes time is part of that very same reasoning. We expand on this in the next section.

2.2 The Idea

Step-logic is proposed as an alternative to the approaches to limited reasoning discussed in Section 2.1.1, where it is *not* a final tray of conclusions in which one is interested, but rather the ever-changing set of conclusions drawn along the way. That is, step-logic is designed to model reasoning that focuses on the on-going process of deduction, where there is no final tray; see Figure 2.2.

The reasoner starts out with an empty set of beliefs at step 0. ‘‘Observations’’ (that is, external inputs to the system) may arise at discrete time-steps. When an observation appears, it is considered a belief. From these beliefs, new beliefs may be concluded. So at some step i the reasoner may have belief α, concluded based on earlier beliefs or arising at step i directly as an observation. Since the process of deduction is part of the same reasoning, these time parameters can figure in the on-going reasoning itself. As an example, we might see $Now(i)$, intuitively ‘‘the time is now i’’, as a conclusion at step i. At some later step j, the reasoner might no longer have the belief $Now(i)$, but would now believe $Now(j)$. The set of beliefs at any step i, together with their sub-formulas, are regarded as the wffs of which the agent is aware at step i.[2]

[2]We deal with this in more detail in Chapter 5.

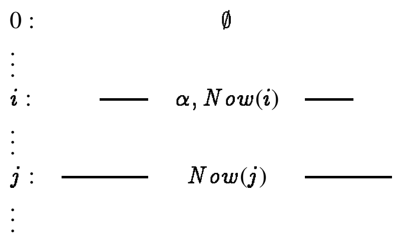

Figure 2.2: Step-like logical studies

We think of each step of reasoning as representing a given fixed interval of time, so that for instance, after 10 steps of reasoning have occurred, so have 10 units of time. Because there are in general a growing number of conclusions at later steps, we would not see this one-to-one correspondence between steps and actual elapsed time in an implementation; the length of time taken to make all deductions for a given step would in general grow in later steps. In our idealization, however, we think of a step as being one unit of time. The long-range focus of this work is to build a ''retraction'' device that would keep the current belief set of reasonable size.[3]

Ordinary logic can be used to model a reasoning agent's activity, but this must be done from *afar*; that is, the logic serves as a meta-theory *about* the agent. If one wanted to have a direct representation of the evolving process of the very reasoning itself, a time argument to a predicate representing the agent's proof process could be added. The most elementary step-logic proposed (SL^0) is just of this sort. However, if we want the *agent* to reason about the passage of time that occurs *as* it reasons, time arguments must be put into the *agent's* language. We could then have such an agent's logic evolving and representing that evolving history at the same time.

There are important differences from ordinary logic, however. Since time goes on as the agent reasons, and since this phenomenon is part of what is to be reasoned about, the agent will need to take note of facts that come and go, e.g., ''It is now 12:00 and I am just starting this task . . . Now it is no longer 12:00, but rather it is 12:15, and I still have not finished the task I began at 12:00.'' This immediately puts us in a non-traditional setting, for we lose monotonicity: as the history evolves, conclusions may be lost. Their loss, however, need not be considered a weakness, but rather a strength, based on a reasoned assessment of a changing situation. It is clear, then, that a step-logic cannot in general retain or inherit all conclusions from one step to the next. The reader is cautioned to keep this in mind in the examples. Despite this feature, we will see that step-logic is primarily a deductive apparatus.

We would like then to construct a logical formalism to serve as a model of a reasoning agent with the ability to reason about the passage of time *as* it is reasoning. Therefore, we must have some way of representing time within the logic which is related to the process of actually producing theorems in that *same* logic.[4] To do this we need to augment the logic with a notion of ''now'', which appropriately changes as deductions are performed. It turns out that this is not an easy task. It is this introduction of a notion of now that makes step-logic critically distinct from traditional logic. While there are many other issues that are related to the general approach being advocated, this thesis concentrates on useful technical devices in time-situated reasoning that pertain specifically to negative introspection.

This new approach to reasoning leads to differences in the meta-theorems that are proved. Those theorems proved about conventional logics are asymptotic in nature, that is, they demonstrate useful properties of the *limiting* case. Many meta-theorems we wish to prove about step-logics, on the other hand, concern the *bounded* case. What is of interest is not that the agent may *eventually* know some α, but rather that it knows α within some given time interval.

[3]See the discussion in Chapter 8.

[4]Could this be done via a Thm predicate; that is, $Thm(i, `\alpha')$ would mean there is a proof of α of length i? No, because having a proof of length i does not force there to be a proof of length $i+1$: α may no longer be a theorem in the next step.

As an example, one of the meta-theorems we prove involves the notion of ‘‘analytic completeness’’. A given meta-theory SL^n is *analytically complete* if for an arbitrary agent formula α, and for any time i,

$$\text{either } SL^n \vdash K(i,\alpha) \qquad \text{or} \qquad SL^n \vdash \neg K(i,\alpha),$$

where $K(i,\alpha)$ is intended to mean that the agent knows α at time i. That is, analytic completeness holds if the meta-theory can prove either that α is an agent theorem at time i, or that α is *not* an agent theorem at time i. A meta-theory with this property is able to determine what the agent does or doesn't know at any given time, and hence can completely analyze the agent's reasoning behavior.[5] Note that we are not concerned here with whether or not the agent will *eventually* know α (what we call an *asymptotic* result), but rather whether α is known *specifically* at time i.

Although the emphasis in this thesis is on the bounded case, certain asymptotic results are sometimes useful and/or interesting. For example, Theorem 4.6 in Section 4.2.2 says that all agent conclusions in SL_0 are tautologies. This is used to arrive at the corollary, which states that a single propositional letter is never proven by the agent. The contra-positive of Theorem 4.6 would be interesting to verify as well, namely, that all tautologies are eventually proven.

2.3 Applications to Commonsense Reasoning

It is easy to provide examples in which the effort or time spent in reasoning is crucial. Three such problems are presented here:

1. The Three-wise-men problem
2. The Brother problem
3. Little Nell

The step-logics investigated so far are not yet powerful enough to solve the third problem. However, detailed solutions, including implementations, are provided for the first two: Chapter 6 contains the solution to the *Brother problem* and Chapter 7 contains the solution to the *Three-wise-men problem*.

2.3.1 The Three-wise-men problem

We present a variation of this classic problem which was first introduced to the AI literature by [McC78]. A king wishes to know whether his three advisors are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down. The wise man in the back (wise man #3) can see the backs of the other two men. The man in the middle (wise man #2) can only see the one wise man in front of him (wise man #1); and the wise man in front (wise man #1) can see neither wise man #3 nor wise man #2. The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the color of his own card. Each wise man must announce the color of his own card as soon as he knows what it is. (The first to correctly announce the color of his own card will be aptly rewarded.) All know that this will happen. The room is silent; then, after several minutes, wise man #1 says ‘‘My card is white!’’.

We assume in this puzzle that the wise men do not lie, that they all have the same reasoning capabilities, and that they can all think at the same speed. We then can postulate that the following reasoning took place. Each wise man knows there is at least one white card. If the cards of wise man #2 and wise man #1 were black, then wise man #3 would have been able to announce immediately that his card was white. They all realize this (they are all truly wise). Since wise man #3 kept silent, either wise man #2's card is white, or wise man #1's is. At this point wise man #2 would be able to determine, if wise man #1's were black, that

[5] SL^0, the only meta-theory that we have constructed, is shown to be analytically complete in Chapter 4.

his card was white. They all realize this. Since wise man #2 also remains silent, wise man #1 knows his card must be white.

The reader should be able to see that it is important in this problem to be able to reason,

> If such and such were true, then so and so *would have realized it by now.*

That is, if wise man #2, for instance, is able to determine that wise man #3 would have been able to figure out by now that wise man #3's card is white, and wise man #2 has heard nothing, then wise man #2 knows that wise man #3 does *not* know the color of his card. Our model of reasoning is particularly well-suited to this type of deduction. See Chapter 7 for a development of and solution to this problem. Others have studied this problem as well (e.g. see [Kon84] and [KL87]), but from a final-tray perspective.

2.3.2 The Brother problem

[Moo83] presents the following problem.

> Consider my reason for believing that I do not have an older brother. It is surely not that one of my parents once casually remarked, ‘‘you know, you don't have any older brothers,’’ nor have I pieced it together by carefully sifting other evidence. I simply believe that if I had an older brother I would surely know about it.

So one must be able to reason, ‘‘since I don't know I have an older brother, I must not.’’

This problem can be broken down into two: the first requires that the reasoner be able to decide he doesn't know he has an older brother; the second that, on that basis, he, in fact, does not have an older brother (from *modus ponens* and the assumption that ‘‘If I had an older brother, I'd know it.’’).

It is easy enough to provide a model of reasoning to account for this latter part of the problem. It is the former part, however, that is more difficult. The reasoner must be able to determine that he doesn't know a particular piece of information. Step-logic, with its finitary introspective capabilities, allows this determination to be made in real-time within the formalism itself. ([Moo83] and [Per88a] have developed formalisms for such reasoning, although both work within a final-tray setting.) The reasoner is able to introspect and determine that he does not have the knowledge of an older brother. With the rule, ‘‘If I have an older brother, I know I have an older brother,’’ the reasoner can then deduce that he, in fact, does not have an older brother. This problem is tackled in detail in Chapter 6, where both a formal solution and an implementation of that solution are given.

2.3.3 Little Nell

Consider Little Nell who has been tied to the railroad tracks. A train is quickly approaching, and Dudley must determine how to save her. [McD82] discusses this problem in terms of prevention of an act---one must prevent Nell from being destroyed by the train. [Haa85] expands upon this work; he distinguishes between *possible* and *actual* situations. What we find pertinent is the fact that Dudley has a *limited amount of time* in which he must find a solution. It is fruitless to go through a lengthy process to determine what should be done, only to find that Nell has meanwhile been mashed.

It is thus important for Dudley to both figure out how to save Nell *and* to do so before it is too late. This requires Dudley to assess how much time he has available, as well as to estimate how long it will take to solve the problem. If his first attempts at a solution seem to him to require more time than appears to be available, he must work on another solution. So, for instance, he might determine that going home to get a pair of scissors in order to cut the rope would be futile, whereas making a call on the emergency phone nearby to alert the engineer is far more reasonable. Dudley must realize that all this must be done as the train is drawing nearer. This then requires Dudley to recognize that his thought processes take time.

This problem is similar to that encountered by a student while taking an exam. The student, knowing there is a limited amount of time in which to answer all questions, must continually assess how many questions he has answered and how many remain, as well as working on actually answering the questions themselves.

Step-logic provides a model for the reasoning needed by the student and Dudley. A particular step-logic modelling this problem, however, is not currently provided.

3 The Details

This chapter describes the technical details of step-logic. Section 3.1 describes the basic components of a step-logic and Section 3.2 contains the precise technical definitions. Several interesting theorems concerning consistency and soundness are also stated and proven.

3.1 Basics

A step-logic is characterized by a language, observations, and inference rules. We emphasize that step-logic is *deterministic* in that at each step i *all* possible conclusions from one application of the rules of inference applied to the previous steps are drawn (and therefore are among the wffs at step i). However, for real-time effectiveness and cognitive plausibility, at each step we want only a finite number of conclusions to be drawn. We would like this set to be not only finite, but quite small. Although the set of conclusions is kept finite, the current formulation of step-logic does not deal with the problem of keeping the set small. Instead we are principally concerned with studying negative introspection and default reasoning in terms of contradiction-handling. That is, *we want step-logic to allow contradictions to arise* as defaults are invoked and yet also *we want the consequences of contradictions to be controlled* in a reasonable manner consistent with commonsense.

We return now to the idea that there are two distinct types of formalisms of interest, that occur in pairs: the meta-theory SL^n *about* an agent, and the agent-theory SL_n itself. Here n is simply an index serving to distinguish different versions of step-logics. It is the latter, SL_n, that is to be step-like; the former, SL^n, is simply our assurance that we have been honest in describing what we mean by a particular agent's reasoning. Thus the meta-theory is to be a scientific theory subject to the usual strictures such as consistency and completeness. The agent theory, on the other hand, may be inconsistent and incomplete; indeed if the agent is an ordinary fallible reasoner it *will* be so. The two theories together form a step-logic *pair*.

We propose three major mechanisms to study as possible aspects of an agent-theory: self-knowledge, time, and retraction. Since it is important for the agent to reason about its own processes, a self, or belief, predicate is needed. We employ a predicate symbol, K, for this purpose: $K(i, `\alpha')$ is intended to mean that the agent knows wff α at time i.[1] K may or may not be part of the agent's own language; however, many kinds of reasoning require that it be. Note that 'α' is a name for α, i.e., a constant term, where the quotes are intended to be quasi-quotes. We drop the quotes in $K(i, `\alpha')$ in the remainder of the dissertation.

In order for the agent to reason about time, a time predicate is needed. This not only amounts to a parameter such as i in $K(i, \alpha)$ as we just saw, but information as to how i relates to the on-going time as deductions are performed. Thus the agent should have information as to what time it is *now*, and this should change as deductions are performed. This feature is what makes step-logic distinctly different from ordinary logic. We use the predicate expression $Now(i)$ to mean the time currently is i. Again, this may or may not be part of the agent's language, but in many cases of interest it is.

Finally, since we want to be able to deal with commonsense reasoning, the agent will have to use default reasoning. That is, a particular fact may be believed if there is no evidence to the contrary; however, later, in the face of new evidence, the former belief may be retracted. For this, we need some kind of a retraction device. Retraction will be facilitated by focusing on the dual: inheritance. We do *not* assume that all deductions at time i are inherited (retained) at time $i+1$. By carefully restricting inheritance we achieve a rudimentary kind of retraction. The most obvious case is that involving the predicate Now: if at a given step the agent knows the time to be i, by having the belief $Now(i)$, then that belief shall not be inherited to the next time step.

[1] We are not distinguishing here between belief and knowledge. See [Get63] for a discussion of belief vs. knowledge.

In [DP86b] we proposed eight step-logic pairs, arranged in increasing sophistication with respect to the three mechanisms above (self-knowledge (S), time (T), and retraction (R)). The agent-theories and the mechanisms that they include are as follows:

- SL_0: none
- SL_1: S
- SL_2: T
- SL_3: R
- SL_4: S, R
- SL_5: S, T
- SL_6: R, T
- SL_7: S, T, R

SL_0 has none of the three mechanisms, and SL_7, the most sophisticated agent-theory, has all three. Of the eight agent-theory/meta-theory pairs, only SL^0 and SL_7, the simplest meta-theory and the most complex agent-theory, have been studied in any detail.[2] The meta-theories are all consistent, first-order theories; however, their associated agent-theories are another matter. These we do not even *want* in general to be consistent, for they are (largely) intended as formal counterparts of the reasoning of fallible agents. SL_0 is an exception, for it, as an initial effort, was constructed to do merely propositional (tautological) reasoning so we could more easily test its meta-theory, SL^0.

3.2 Technical Definitions

In order to precisely define step-logic, we now present several definitions.

Intuitively, we view an agent as an inference mechanism that may be given external inputs or observations. Inferred wffs are called beliefs; these may include certain observations.

Let $\mathcal{L}$ be a first-order or propositional language, and let $\mathcal{W}$ be the set of wffs of $\mathcal{L}$.

Definition 3.1 *An* observation-function *is a function $OBS : \mathbf{N} \rightarrow \mathcal{P}(\mathcal{W})$, where $\mathcal{P}(\mathcal{W})$ is the powerset of $\mathcal{W}$, and where for each $i \in \mathbf{N}$, the set $OBS(i)$ is finite. If $\alpha \in OBS(i)$, then α is called an i-*observation.

Definition 3.2 *A* history *is a finite tuple of pairs of finite subsets of $\mathcal{W}$. $\mathcal{H}$ is the set of histories.*

Definition 3.3 *An* inference-function *is a function $INF : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{W})$, where for each $h \in \mathcal{H}$, $INF(h)$ is finite.*

Intuitively, a history is a conceivable temporal sequence of belief-set/observation-set pairs. The history is a *finite* tuple; it represents the temporal sequence up to a certain point in time. The inference-function extends the temporal sequence of belief sets by one more step beyond the history. In the example in Figure 3.1 we see that these ideas are used to generate an actual history based on an inference-function and an observation-function. Definitions 3.4 and 3.5 formalize this in terms of a step-logic SL_n.

Definition 3.4 *An SL_n-*theory over a language $\mathcal{L}$ *is a triple, $< \mathcal{L}, OBS, INF >$, where $\mathcal{L}$ is a first-order language, OBS is an observation-function, and INF is an inference-function. We use the notation, $SL_n(OBS, INF)$, for such a theory (the language $\mathcal{L}$ is implicit in the definitions of OBS and INF). If we wish to consider a fixed INF but varied OBS, we write $SL_n(\cdot, INF)$.*

Let $SL_n(OBS, INF)$ be an SL_n-theory over $\mathcal{L}$.

[2]We describe SL^0 in Chapter 4 and SL_7 in Chapter 5.

Let

- $OBS(i) = \begin{cases} \{bird(x) \to flies(x)\} & \text{if } i = 1 \\ \{bird(tweety)\} & \text{if } i = 3 \\ \emptyset & \text{otherwise} \end{cases}$
- $Thm_i \subseteq \mathcal{W}$, $0 \leq i < n$; $Thm_0 = \emptyset$;
- $INF(<< Thm_0, OBS(1) >, \ldots, < Thm_{n-1}, OBS(n) >>) =$
  $Thm_{n-1} \cup OBS(n) \cup \{\alpha(t) \mid (\exists\beta)(\beta(t), \beta(x) \to \alpha(x) \in (Thm_{n-1} \cup OBS(n)))\}$.

The history h of the first five steps then would be:

$$\begin{array}{rlcl} h = & << & \emptyset & ,\{bird(x) \to flies(x)\}>, \\ & < & \{bird(x) \to flies(x)\} & , \quad \emptyset \quad >, \\ & < & \{bird(x) \to flies(x)\} & , \{bird(tweety)\} >, \\ & < & \{bird(x) \to flies(x), bird(tweety), flies(tweety)\}, & \emptyset \quad >, \\ & < & \{bird(x) \to flies(x), bird(tweety), flies(tweety)\}, & \emptyset \quad >> \end{array}$$

Figure 3.1: Example of OBS and INF

Definition 3.5 *Let the set of 0-theorems, denoted Thm_0, be empty. For $i > 0$, let the set of i-theorems, denoted Thm_i, be $INF(< < Thm_0, OBS(1) >, < Thm_1, OBS(2) >, \ldots, < Thm_{i-1}, OBS(i) > >)$. We write $SL_n(OBS, INF) \vdash_i \alpha$ to mean α is an i-theorem of $SL_n(OBS, INF)$.*[3]

Definition 3.6 *Given a theory $SL_n(OBS, INF)$, a corresponding SL^n-*theory*, written $SL^n(OBS, INF)$, is a first-order theory having binary predicate symbol K,*[4] *numerals, and names for the wffs in $\mathcal{L}$, such that*

$$SL^n(OBS, INF) \vdash K(i, \alpha) \ \ \textit{iff} \ \ SL_n(OBS, INF) \vdash_i \alpha.$$

Thus in $SL^n(OBS, INF)$, $K(i, \alpha)$ is intended to express that α is an i-theorem of $SL_n(OBS, INF)$.

A notion of completeness for a meta-theory can be defined as follows:

Definition 3.7 *A meta-theory SL^n is* analytically complete, *if for every positive integer i, and every constant α naming an agent wff of the corresponding agent-theory, either $SL^n \vdash K(i, \alpha)$ or $SL^n \vdash \neg K(i, \alpha)$.*

In Chapter 4 we show that SL^0 is in fact analytically complete. But what kind of completeness might be wanted for an *agent*-theory? In SL_0, it is desirable that every tautology be (eventually) provable. This is the case, since every tautology has a proof in propositional logic and, for a sufficiently large value of i, all axioms (i.e., the ''observations'') in such a proof will have appeared (by design of SL_0) by step i. Thus SL_0 is complete with respect to the intended domain, namely, tautologies. However, for other step-logics the case is not so simple, for the intended domain, namely, the commonsense world, has no well-understood precise definition. Nevertheless, we can isolate special cases in which certain meta-theorems are possible. In particular, if no non-logical axioms (beliefs) are given to an agent at step 0 (or any later time), then it is reasonable to expect the agent to remain consistent. This we establish for SL_7 in which the logical axioms do not contain the predicate symbol ''Now.''

Let $\mathcal{L}'$ be the language having the symbols of $\mathcal{L}$ and the (possibly additional) predicate symbols K and Now. Thus $\mathcal{L}'$ may be $\mathcal{L}$ itself.

Definition 3.8 *A* step-interpretation *for $\mathcal{L}'$ is a sequence $M = < M_0, M_1, \ldots, M_i, \ldots >$, where*

1. *Each M_i is an ordinary first-order interpretation of $\mathcal{L}'$.*
2. *$M_i \models Now(i)$.*

[3]Note the non-standard use of the turnstile here.

[4]We see that the predicate letter K has two roles: in SL^n and in SL_n. The context will make the role clear.

Definition 3.9 *A* step-model *for $SL_n(OBS, INF)$ is a step-interpretation M satisfying*

1. *$M_i \models K(i, \alpha)$ iff $SL_n(OBS, INF) \vdash_i \alpha$.*
2. *$M_i \models \alpha$ whenever $SL_n(OBS, INF) \vdash_i \alpha$.*

Condition 1 insures that an historical record of the i-theorems exists; and Condition 2 insures that the i-theorems are in fact true.

Definition 3.10 *A wff α is i-*true *in a step-model M (written $M \models_i \alpha$) if $M_i \models \alpha$.*

Definition 3.11 *$SL_n(OBS, INF)$ is* step-wise consistent *if for each $i \in \mathbf{N}$, the set of i-theorems is consistent.*

Definition 3.12 *$SL_n(OBS, INF)$ is* eventually consistent *if $\exists i$ such that $\forall j > i$, the set of j-theorems is consistent.*

Definition 3.13 *An observation-function OBS is* finite *if $\exists i$ such that $\forall j > i$, $OBS(j) = \emptyset$.*

Definition 3.14 *$SL_n(\cdot, INF)$ is* self-stabilizing *if for every finite OBS, $SL_n(OBS,\ INF)$ is eventually consistent.*

Remark 3.15 *Note the following:*

1. *Even if $SL_n(OBS, INF)$ is step-wise consistent, it can have conflicting wffs at* different *steps, e.g., $SL_n(OBS, INF) \vdash_{10} Now(10)$ and $SL_n(OBS, INF) \vdash_{11} \neg Now(10)$.*
2. *Any step-wise consistent theory is eventually consistent.*
3. *Intuitively a self-stabilizing theory $SL_n(\cdot, INF)$ corresponds to a fixed agent that can regain and retain consistency after being given arbitrarily (but finitely) many contradictory initial beliefs.*

Theorem 3.16 *If $SL_n(OBS, INF)$ has a step-model, then it is step-wise consistent.*

Proof: Let $SL_n(OBS, INF)$ have a step-model $M = < M_0, M_1, \ldots, M_i, \ldots >$. Let $j \in \mathbf{N}$ be arbitrary. Then for each α in the set of j-theorems, $M_j \models \alpha$. This means that the set of j-theorems is consistent, since it has a (standard first-order) model M_j. $\square$

Theorem 3.17 (Soundness) *Every step-logic $SL_n(OBS, INF)$ is sound with respect to step-models. That is, every i-theorem α of $SL_n(OBS, INF)$ is i-true in every step-model M of $SL_n(OBS, INF)$, i.e., if $SL_n(OBS, INF) \vdash_i \alpha$ then $M \models_i \alpha$.*

Proof: Let α be an i-theorem of $SL_n(OBS, INF)$, and let M be a step-model of $SL_n(OBS, INF)$. $SL_n(OBS, INF) \vdash_i \alpha$, so by definition of step-model, $M_i \models \alpha$, and hence (by definition of i-true) $M \models_i \alpha$. $\square$

3.3 Summary

In this chapter we described the technical details of step-logic. Precise definitions were given and several theorems concerning the consistency and soundness of step-logics were stated and proven.

4 $\mathbf{SL_0}$ and $\mathbf{SL^0}$

We turn now to a specific step-logic pair: $< SL_0, SL^0 >$. This is the simplest of the step-logic pairs. The agent's theory, SL_0, is merely propositional logic. The meta-theory, SL^0, however, is based on the idea that reasoning occurs in steps.

4.1 Introduction

We reiterate the sharp distinction between the *agent's* language and theory (SL_0), and *our* (the scientist's) language and theory (SL^0). The agent has one set of symbols, axioms, and rules, while we have another. SL_0 has neither self-knowledge (S), time (T), nor retraction (R). To simplify it even more, SL_0 contains only propositions; it has no variables. (For definiteness, we have arbitrarily selected one of the standard axiomatizations in the literature.) As such, then, SL^0 is basically a formalism to help us, as scientists, to understand the reasoner. It does not allow the agent to do any reasoning about his own reasoning.

As before, we use the notation $K(i, \alpha)$ to indicate that the agent has proven α in i steps. ''α'' is any formula in the *agent's* language, but is treated as a constant in SL^0. In SL^0, the ''i'' is just *our* notation; the agent has no way of knowing that it is at time i that it has proven α (it just ''knows'' α at time i). For example, it might be the case that we have been able to prove that the agent has been unable to prove ''P'' in time i, where P is a predicate letter in the agent's language. (This in particular would be the case for an agent using ordinary propositional logic.) We write this $\vdash \neg K(i, P)$.

We will continue the convention of using Greek letters for the agent's wffs. These also serve as terms of SL^0. To further distinguish SL^0 and SL_0, we use ''implies'' and ''not'' as function symbols of SL^0 to designate implication and negation, respectively, of the agent's wffs. The goal is to design SL^0 to be strong enough so that it is analytically complete, that is, for each $i \in \mathcal{N}$, and for each $\alpha \in \mathcal{L}(SL_0)$ (where $\mathcal{L}(SL_0)$ is the agent's language), we can say

$$\text{either } SL^0 \vdash K(i, \alpha) \qquad \text{or} \qquad SL^0 \vdash \neg K(i, \alpha).$$

We have been able to achieve analytic completeness for propositional agents; that is, for any wff α in the agent's language and any time i, SL^0 can either prove $K(i, \alpha)$ or can prove its negation. This means that the agent's reasoning over time is completely characterized, in the sense that we know what it has and has not established at each moment. As might be expected, the difficult part is determining when a wff of $\mathcal{L}(SL_0)$ is *not* deduced after i steps. (Intuitively, showing $K(i, \alpha)$ requires exhibiting a single proof of α, whereas showing $\neg K(i, \alpha)$ requires exhibiting that no proof of α exists.)

We now sketch the intuitive operation of our intended agent. For each wff α in the agent's language $\mathcal{L}(SL_0)$, and for each time-step $i \in \mathcal{N}$, $K(i, \alpha)$ is to hold (i.e., the agent has deduced α by time i) if and only if there is a formal proof of α in i or fewer steps using only *modus ponens* and whatever axioms can be retrieved in i steps. Here axioms are conceived to be constructed by the agent little by little. That is, the agent begins with no axioms at all, and at any step i has access to all old conclusions as well as any axioms using no more than the first i proposition letters and no more than i connectives (instances of $\neg$ and $\rightarrow$).

We must ''feed in'' the axioms a few at a time because we do not want the agent to have an infinite number of axioms at any given step. We can say then that the agent is ''aware of'' only certain axioms at any given step. This has some similarity to Fagin and Halpern's notion of awareness (see [FH88] and Section 2.1.1).

In order to distinguish the agent's wffs from our own (i.e., wffs of SL_0 from wffs of SL^0), we write $not(\alpha)$ for the agent's negation of α, and $implies(\alpha, \beta)$ for the agent's representation of $(\alpha \rightarrow \beta)$. In SL^0, then, *not* and *implies* become function symbols. Also, for an agent's wff to appear as a constant term in SL^0, proposition letters must be constant symbols of $\mathcal{L}(SL^0)$.

4.2 The Theory

4.2.1 Axioms

SL^0 is given enough arithmetic to reason about steps as integers. In addition, the following principal axioms and axiom schemata are given.

AX: $(\forall\alpha)[Ax(\alpha) \longleftrightarrow$
$[(\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma,\beta)))\vee$
$(\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)), implies(implies(not(\gamma),\beta),$
$\gamma)))\vee$
$(\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma,\delta)), implies(implies(\beta,\gamma),$
$implies(\beta,\delta))))]]$

AX says that axioms are the usual three tautology types, as in [Men87].

THM: $(\forall\alpha)(\forall i)(K(i,\alpha) \longleftrightarrow [Feed(\alpha,i) \vee Mp(\alpha,i)])$

*THM defines what it means for α to be proven by the agent at time-step i: either α has been ''fed in'' ($Feed(\alpha,i)$), or α has been derived through *modus ponens* from previous steps ($Mp(\alpha,i)$).*

ALP: $(\forall\alpha)(\forall i)[Feed(\alpha,i) \longleftrightarrow (Ax(\alpha) \wedge l(\alpha) \leq i \wedge p(\alpha) < i)]$

Those wffs which are fed in at time-step i are exactly the axioms of length less than or equal to i, with a maximal index strictly less than i.

MP: $(\forall\alpha)(\forall i)[Mp(\alpha,i) \longleftrightarrow (\exists j)(\exists k)(\exists\beta)(K(j,\beta) \wedge K(k, implies(\beta,\alpha)) \wedge (j < i) \wedge (k < i))]$

*This is a version of *modus ponens*.*

FEED: $(\forall\alpha)[Feed(\alpha,i) \longleftrightarrow (\alpha = \phi_1 \vee \alpha = \phi_2 \vee \ldots \vee \alpha = \phi_{l_i})]$, for all $i \geq 2$.

This lists exactly those axioms that are fed in at time-step i. ϕ_1 thru ϕ_{l_i} are wffs in $\mathcal{L}(SL_0)$ that are axioms (according to AX), have length(i.e., number of connectives) less than or equal to i, and have a maximal index of i. Note that the number of axioms that are fed in monotonically increases, i.e., l_{i+1} is greater than l_i, for all i.

TABULARASA: $(\forall\alpha)(\forall i)[K(i,\alpha) \rightarrow i \geq 0]$

The agent knows nothing before time step 0.

TFCN1: $(\forall x)[Tfcn(x) \rightarrow$
$(\forall\alpha)(\forall\beta)[True(x, implies(\alpha,\beta)) \longleftrightarrow (True(x,\beta) \vee \neg True(x,\alpha))]]$

TFCN2: $(\forall x)[Tfcn(x) \rightarrow (\forall\alpha)[True(x, not(\alpha)) \longleftrightarrow \neg True(x,\alpha)]]$

TFCN1 and TFCN2 say that truth-functions behave properly with respect to connectives.

TAUT1: $(\forall\alpha)[Taut(\alpha) \longleftrightarrow (\forall x)[Tfcn(x) \rightarrow True(x,\alpha)]]$

TAUT2: $(\forall\alpha)[Ax(\alpha) \rightarrow Taut(\alpha)]$

TAUT1 and TAUT2 say, respectively, that tautologies are wffs that are true under all truth-functions, and that axioms are tautologies.

PL1: $Pl(P_i)$, for all $i \in \mathcal{N}$.

PL2: $(\forall\alpha)[Pl(\alpha) \rightarrow (\exists x)[Tfcn(x) \wedge \neg True(x,\alpha)]]$

*PL1 says that $P_0, P_1, P_2, \ldots$ are propositional letters. PL2 says that for each propositional letter there is a truth-function making it false. Note that this does not give *all* the usual truth-functions, but it is sufficient for our purposes.*

LN1: $(\forall\alpha)(Pl(\alpha) \longleftrightarrow l(\alpha) = 0)$

LN2: $(\forall\alpha)[l(not(\alpha)) = l(\alpha) + 1]$

LN3: $(\forall\alpha)(\forall\beta)[l(implies(\alpha,\beta)) = l(\alpha) + l(\beta) + 1]$

LN4: $(\forall\alpha)(l(\alpha) \geq 0)$

The function $l(\alpha)$ returns the length of α, i.e., the number of connectives in α(this being 0 for propositional letters).

P1: $(\forall\alpha)(\forall i)[(Pl(\alpha) \wedge p(\alpha) = i) \longleftrightarrow \alpha = P_i]$

P2: $(\forall\alpha)[p(not(\alpha)) = p(\alpha)]$

P3: $(\forall\alpha)(\forall\beta)[p(implies(\alpha,\beta)) = max(p(\alpha), p(\beta))]$

P4: $(\forall\alpha)[p(\alpha) \geq 0]$

The function $p(\alpha)$ returns the maximum index of all the propositional letters in α.

MAX1: $(\forall i)(\forall j)[i \geq j \longleftrightarrow max(i,j) = i]$

MAX2: $(\forall i)(\forall j)[max(i,j) = max(j,i)]$

The function max returns the maximum of its two arguments.

EQ1: $s \neq t$, for all distinct propositional letters $s, t \in \mathcal{L}(SL_0)$.

EQ2: $(\forall\alpha)(\forall\beta)(\forall\gamma)[Pl(\alpha) \rightarrow \alpha \neq implies(\beta,\gamma)]$

EQ3: $(\forall\alpha)(\forall\beta)[Pl(\alpha) \rightarrow \alpha \neq not(\beta)]$

EQ4: $(\forall\alpha)(\forall\beta)(\forall\gamma)[implies(\alpha,\beta) \neq not(\gamma)]$

EQ5: $(\forall\alpha)(\forall\beta)(\forall\gamma)(\forall\delta)[implies(\alpha,\beta) = implies(\gamma,\delta) \longleftrightarrow \alpha = \gamma \wedge \beta = \delta]$

EQ6: $(\forall\alpha)(\forall\beta)[not(\alpha) = not(\beta) \longleftrightarrow \alpha = \beta]$

EQ1, EQ2, and EQ3 say that distinct agent wffs represent distinct objects. EQ4 says that a wff whose main connective is an implication arrow cannot be equal to a wff whose main connective is a negation symbol. EQ5 (EQ6) says that in order for two wffs to be equal, where the implication arrow(negation symbol)is the main connective, their arguments must be equal.

4.2.2 Theorems

The following are some of the interesting results which were obtained for SL^0.

Theorem 4.1 (Analytic Completeness Theorem) *For each $i \in \mathcal{N}$, and for each $\alpha \in \mathcal{L}(SL_0)$,*

$$\text{either } SL^0 \vdash K(i,\alpha) \qquad \text{or} \qquad SL^0 \vdash \neg K(i,\alpha).$$

SL^0 can characterize exactly what has and has not been proved at any given time i.

Lemma 4.2 (Newborn Lemma) $SL^0 \vdash (\forall\alpha)(\neg K(0,\alpha))$.
The agent initially can prove nothing.

Lemma 4.3 (Monotonicity Lemma) $SL^0 \vdash (\forall i)(\forall\alpha)[K(i,\alpha) \rightarrow K(i+1,\alpha)]$.
Once a belief is held, it remains.

Lemma 4.4 (Boundedness Lemma) *Let $i \in \mathcal{N}$, $i \geq 2$. Then $\exists\alpha_1 \ldots \alpha_{n_i} \in \mathcal{L}(SL_0)$, such that*

$$SL^0 \vdash (\forall\alpha)[K(i,\alpha) \longleftrightarrow (\alpha = \alpha_1 \vee \ldots \vee \alpha = \alpha_{n_i})].$$

Theorem 4.5 $SL^0 \vdash (\forall\alpha)(\forall\beta)[[Taut(\alpha) \wedge Taut(implies(\alpha,\beta))] \rightarrow Taut(\beta)]$.

Theorem 4.6 *For any $i \in \mathcal{N}$, $SL^0 \vdash (\forall\alpha)[K(i,\alpha) \rightarrow Taut(\alpha)]$.*
The only wffs an agent can prove are those that are tautologies.

Lemma 4.7 $SL^0 \vdash \neg Taut(P_i)$*, for any propositional letter $P_i \in \mathcal{L}(SL_0)$.*
P_i is not a tautology, for all propositional letters P_i.

Corollary 4.8 $SL^0 \vdash \neg K(i,P_j)$*, for any $i, j \in \mathcal{N}$.*
The agent can never prove P_j, where P_j is a propositional letter.

Proofs of the above results tend to be rather long, and proceed mainly by induction on i and/or the number of connectives in α. The details of the proofs can be found in Appendix A.

4.3 Implementation of SL_0

SL_0 has been implemented on an IBM PC-AT using Arity Prolog. The agent's propositions are represented as $p(0), p(1), p(2), \ldots$ Valid formulas are built up from the propositions using neg and imp in the expected way. $K(i, \alpha)$ is represented as $state(i, \alpha)$. As anticipated, the only formulas that are derived are those that are tautologies, since the modelled agent is a propositional reasoner. One can query the system to determine, among other things, when a given formula is derived. One can also ask for all the formulas that are derived in a given step.

It was hoped that PROLOG, being a *logic* programming language, would work well as the implementation tool for SL_0. It turns out that because PROLOG uses depth-first search, it is rather tricky to get the axioms of SL_0 fed in as desired. It was necessary to alter slightly the order in which the axioms are fed in. (It is still based on the formula's length and the maximum subscript value within the formula, however.)

The program runs more slowly than desired. This is directly related to the manner in which $K(i, \alpha)$ is defined in SL^0. It may be that with suitable programming ''tricks'', the runtime speed can be drastically improved.

PROLOG is also sufficient as the meta description of subsequent step-logics. We have been able to implement a version of SL_7, and use it to provide a solution to the *Brother problem* and the *Three-wise-men problem*.

Several example queries can be found in Appendix C. The PROLOG program is in Appendix D.

4.4 Summary

In this chapter we presented the simplest step-logic pair, $< SL_0, SL^0 >$. The agent-theory, SL_0, has none of the three mechanisms that we postulate are necessary to do commonsense reasoning (self-knowledge, time, and retraction). It, in fact, models a mere propositional reasoner. Its intuitive operation is sketched. The meta-theory is then formally defined. It has all the apparatus needed to describe the agent's proof process.

Meta-theorems about SL^0 were presented, the most important of which is *Analytic Completeness*. This theorem states that for any wff α and any time-step i, SL^0 can prove that either α has indeed been proven by the agent by step i, or that in fact the agent has not yet been able to prove α by step i. The fact that we were able to prove this theorem indicates that we have been successful in building a formalism that correctly characterizes the agent. Proofs of all the theorems can be found in Appendix A.

SL_0 has been implemented. A sample run can be found in Appendix C.

Indeed, $< SL_0, SL^0 >$ is not very interesting, as it does not allow the agent to do any reasoning about its own reasoning. It was studied, however, to help us as scientists to understand the reasoner (which in this case happened to be a very simple reasoner).

5 $\mathbf{SL_7}$

In this chapter we describe what is so far the most ambitious step-logic: SL_7. Unlike SL_0, SL_7 is intended not for derivation of tautologies but rather for the study of particular default capabilities; in particular, tautologies and other logical axioms are not generally employed in SL_7. We use the notation SL_7 for any of a family of step-logics whose OBS and INF involve the predicates Now and K and contain a retraction mechanism. Choosing OBS and INF therefore fixes the theory within the family.

5.1 Introduction

SL_7, as stated earlier, is *not* intended in general to be consistent. If supplied *only* with logically valid wffs that are Now-free on which to base its reasoning, then indeed SL_7 will remain consistent over time: there will be no step i at which the conclusion set is inconsistent, for its rules of inference are sound (see Theorem 5.4 in Section 5.2). However, virtually all the interesting applications of SL_7 involve providing the agent with some non-logical and potentially false axioms, thus opening the way to derivation of contradictions. In traditional logics, the introduction of a contradiction immediately introduces all other wffs into the system as theorems. We refer to this phenomenon as the *swamping problem*. The controlled growth of deductions in step-logic provides a convenient tool for avoiding the swamping problem. This behavior is what we are interested in studying.

5.2 The Theory

The language of SL_7 is first-order, having unary predicate symbol, Now, binary predicate symbol, K, and ternary predicate symbol, $Contra$, for time, knowledge, and contradiction, respectively. We write $Now(i)$ to mean the time is now i; $K(i, \alpha)$ can be thought of as stating that α is known[1] at step i; and $Contra(i, \alpha, \beta)$ means that α and β are in direct contradiction (one is the negation of the other) and both are i-theorems. Note that K is used here as a predicate of the *agent*-theory. Context should make clear whether we are talking about the agent- or meta-theory.

The formulas that the agent has at step i (the i-theorems) are precisely all those that can be deduced from step $i - 1$ using one application of the applicable rules of inference. As previously stated, the agent is to have only a finite number of theorems (conclusions, beliefs, or simply wffs) at any given step. We write:

$$\begin{array}{rl} i: & \alpha \\ i+1: & \beta \end{array}$$

to mean that α is an i-theorem, and β is an $i+1$-theorem. There is no implicit assumption that α (or any other wff other than β) is present (or not present) at step $i+1$. Wffs are not assumed to be inherited or retained in passing from one step to the next, unless explicitly stated in an inference rule. In Figure 5.1 on page 20, we illustrate one possible inference function, denoted INF_B, involving a rule for special types of inheritance; see Rule 7.

SL_7 has all three mechanisms that we have proposed: time, self-knowledge, and retraction. For *time*, we envision a clock which is ticking as the agent is reasoning. At each step in its reasoning, the agent looks at this clock to obtain the time. The wff $Now(i)$ is an i-theorem. $Now(i)$ corresponds intuitively to the statement ''The time is now i.''

Self-knowledge involves the predicate K, and (in INF_B) a new rule of inference, namely a rule of (negative) introspection; see Rule 5 in Figure 5.1 below. This rule is intended to have the following effect.

[1]known, believed, or concluded. The distinctions between these (see [Get63, Per86, Per88b]) will not be addressed here.

$\neg K(i, \alpha)$ is to be deduced at step $i + 1$ if α is not an i-theorem, but does appear as a closed sub-formula at step i.[2] We regard the closed sub-formulas at step i as approximating the wffs that the agent is ''aware of'' at i.[3] Thus the idea is that the agent can tell at $i + 1$ that a given wff it is *aware* of at step i is not one of those it has as a *conclusion* at i. We will use the K concept to allow the agent to negatively introspect, i.e., to reason at step $i + 1$ that it did not know β at step i. Thus, using INF_B, if α and $\alpha \rightarrow \beta$ are i-theorems, then β and $\neg K(i, \beta)$ will be $i + 1$-theorems (concluded via Rules 3 and 5, respectively). Currently we do not employ positive introspection (i.e., from α at i infer $K(i, \alpha)$ at $i + 1$), although it can be recaptured from axioms if needed.

Retractions are used to facilitate removal of certain conflicting data. Handling contradictions in a system of this sort can be quite tricky. The fact that contradictions *can* arise without upsetting the system (that is, without causing the swamping problem) is a major focus of this work. Currently we handle contradictions by simply not inheriting the formulas directly involved. In future work we hope to have a more sophisticated mechanism. See Section 8.2 for a discussion of this.

$SL_7(\cdot, INF_B)$ was formulated with applications such as the *Brother problem* (see Chapter 6) in mind. This led to the rules of inference listed in Figure 5.1. Rule 3 states, for instance, that if α and $\alpha \rightarrow \beta$ are i-theorems, then β will be an $i + 1$-theorem. Rule 3 makes no claim about whether or not α and/or $\alpha \rightarrow \beta$ are $i + 1$-theorems. The axioms (i.e., the ''observations'') are listed in Chapter 6, where we discuss specifically, the *Brother problem*.

The inference rules given here correspond to the inference-function INF_B. For any given history, INF_B returns the set of all immediate consequences of Rules 1--7 applied to the last step in that history. Recall from page 19 that there is no implicit assumption that wffs are inherited from one step to the next; this must be explicitly stated in an inference rule. Note that Rule 5 is the only default rule.

Rule 1 :	$\dfrac{i:}{i+1 : Now(i+1)}$	Agent looks at clock
Rule 2 :	$\dfrac{i:}{i+1 : \alpha}$	If $\alpha \in OBS(i+1)$
Rule 3 :	$\dfrac{i : \alpha, \alpha \rightarrow \beta}{i+1 : \beta}$	Modus ponens
Rule 4 :	$\dfrac{i : P_1 a, \ldots, P_n a, (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Qx]}{i+1 : Qa}$	Extended modus ponens
Rule 5 :	$\dfrac{i:}{i+1 : \neg K(i, \beta)}$	Negative introspection[a]
Rule 6 :	$\dfrac{i : \alpha, \neg\alpha}{i+1 : Contra(i, \alpha, \neg\alpha)}$	Contradiction is noted
Rule 7 :	$\dfrac{i : \alpha}{i+1 : \alpha}$	Inheritance[b]

[a] where β is not an i-theorem, but is a closed sub-formula at step i.

[b] where nothing of the form $Contra(i-1, \alpha, \beta)$ nor $Contra(i-1, \beta, \alpha)$ is an i-theorem, and where α is not of the form $Now(\beta)$. That is, direct contradictions and time are not inherited.

Figure 5.1: Rules of inference corresponding to INF_B

Unlike SL_0 which is monotonic (that is, if α is an i-theorem, then α is also an $i + 1$-theorem), SL_7

[2] A sub-formula of a wff is any consecutive portion of the wff that itself is a wff. Note that there are only finitely many such sub-formulas at any given step. Rule 5 formalizes the introspective time-delay discussed in Section 3.1.

[3] ''You can't know you don't know something, if you never heard of it.'' Fagin and Halpern ([FH88]) have a different treatment of awareness. See Section 2.1.1 for a discussion of their notion.

is non-monotonic. In $SL_7(\cdot, INF_B)$, a conclusion in a given step i, is inherited to step $i+1$ if it is not contradicted at step i and it is not the predicate $Now(j)$, for some j; see Rule 7 in Figure 5.1. The intuitive reason time is not inherited is that time changes at each step. The intuitive reason contradicting wffs α and β are not inherited is that not both can be true, and so the agent should, for that reason, be unwilling to simply assume either to be the case without further justification. This does not mean, however, that neither will appear at the next step, for either or both may appear for other reasons, as will be seen. Note also that the wff $Contra(i, \alpha, \neg\alpha)$ *will* be inherited, since it is not itself either time or a contradiction, and (intuitively) it expresses a fact (that there was a contradiction at step i) that remains true.

Note that central to our approach is the idea that, for at least some conclusions that our agent is to make, the time the conclusion is drawn is important. For instance, if it concluded at time (step) 5 that B is unknown, we prefer the agent to conclude $\neg K(5, B)$ rather than simply $\neg K(B)$. The reason for this is that it may indeed be true that B is unknown at time 5, but that later B becomes known; this latter event however should not force the agent to forget the (still true) fact that *at time* 5, B was unknown. On the other hand, if we put time stamps on *all* conclusions, then B itself, once concluded, will require more complex inheritances in order to carry B on from step to step as a continuing truth. Thus it seems preferable not to time-stamp every conclusion. This leaves us with the problem of deciding which conclusions to stamp; currently we are stamping only introspections, contradictions, and ''clock look-ups''.

It is worth amplifying on the use of Contra. Suppose that at step i the agent has the wffs $\neg\alpha$, $\neg\beta$, and $\alpha \vee \beta$. (They are all i-theorems.) While these are indeed mutually inconsistent, they do not form a *direct* contradiction; it takes some further work to see the contradiction. If, for instance, at step $i+1$ the agent deduces β (say, from a further wff $\neg\alpha \wedge (\alpha \vee \beta) \rightarrow \beta$ also present at step i), then at step $i+1$ there would be a direct contradiction. This would then be noticed (via Rule 6) at step $i+2$ with the wff $Contra(i+1, \beta, \neg\beta)$. Then (by Rule 7) neither β nor $\neg\beta$ would be inherited to step $i+3$. Note that what is not inherited is context-dependent: if a slightly different line of reasoning had led from the same wffs at step i to a different contradiction at $i+1$, different wffs would fail to be inherited. Thus it is the actual time-trace of past reasoning that is reflected in the decision as to what wffs to distrust. Also note that if the extra wff that allowed the implicit contradiction to become direct had not been present, the implicit contradiction might have remained indefinitely. This behavior we regard as within the spirit of the reasoning we wish to study, since it follows real-time vagaries of what is actually done rather than an externally proscribed notion of validity.

We define several terms before presenting a theorem about the consistency of $SL_7(\cdot, INF_B)$.

Definition 5.1 *A wff is said to be* P-free *if it does not contain the predicate letter* P.

Definition 5.2 *An observation-function OBS is said to be* P-free *if* $\forall i \forall \alpha (\alpha \in OBS(i) \rightarrow \alpha$ *is P-free*).

Definition 5.3 *An observation-function OBS is said to be* valid *if* $\forall i \forall \alpha (\alpha \in OBS(i) \rightarrow \alpha$ *is logically valid*).

Theorem 5.4 *$SL_7(OBS, INF_B)$ is step-wise consistent if OBS is both valid and Now-free.*

Proof: See Appendix B for the details. The idea is to show $SL_7(OBS, INF_B)$ has a step-model, and apply Theorem 3.16 (see page 14). □

Remark 5.5 *When OBS is empty (i.e. $\forall i, OBS(i) = \emptyset$), $SL_7(OBS, INF_B)$ reduces to a ''clock'', i.e. $\forall i$, $SL_7(OBS, INF_B) \vdash_i \alpha$ iff $\alpha = Now(i)$.*

5.3 Implementation of SL_7

SL_7, as stated earlier, represents a family of step-logics. Fixing the observation- and inference-functions fixes the theory. Implementations and sample runs are given for three of these theories. See Appendix C for the example runs and Appendix D for the PROLOG code.

5.4 Summary

In this chapter we presented the most sophisticated agent-theory, SL_7. SL_7 has all three of the mechanisms that we postulate are necessary to do commonsense reasoning: self-knowledge, time, and retraction.

SL_7 actually represents a family of step-logics. The step-logics differ in the particular observation- and inference-functions that are used in the definition of the specific step-logic. In this chapter we described one particular inference-function, which was designed with the *Brother problem* in mind.

A meta-theorem about the consistency of particular SL_7-theories was stated and proved.

SL_7 is far more interesting than the agent-theory SL_0 which was described in the previous chapter. SL_7 is a useful model for several types of commonsense reasoning problems. Because it allows the agent to introspect on his own beliefs *within the formalism itself*, and because contradictions are allowed to occur, a computationally tractable solution to default reasoning becomes possible.

Chapters 6 and 7 define particular SL_7-theories which are suitable for the *Brother problem* and the *Three-wise-men problem*, respectively.

6 The Brother Problem

In this chapter I report on work done in [EDP88] in which we presented a real-time solution to the *Brother problem* (see [Moo83]). An implementation for this problem can be found in Appendix D.

6.1 Statement of the Problem

We reiterate the problem which was described above in Section 2.3. Moore explains:

> Consider my reason for believing that I do not have an older brother. It is surely not that one of my parents once casually remarked, ''you know, you don't have any older brothers,'' nor have I pieced it together by carefully sifting other evidence. I simply believe that if I had an older brother I would surely know about it.

So one must be able to reason, ''since I don't know I have an older brother, I must not.'' We break this problem into two parts: the first requires that the reasoner be able to decide he doesn't know he has an older brother; the second that, on that basis, he, in fact, does not have an older brother. This latter part requires *modus ponens* and the assumption ''If I had an older brother, I'd know it.'' The former part, however, is not so straightforward. It requires the reasoner to determine that he doesn't know something. Step-logic allows this negative reflection within the logic itself, and thus can provide a convincing model for this problem.[1]

In the next section we present three different scenarios. In each the agent must determine whether or not an older brother exists. Let B be a 0-argument predicate letter representing the proposition that an older brother exists. Let P be a 0-argument predicate letter (other than B) that represents a proposition that implies that an older brother exists.[2] In each case, at some step i the agent has the axiom $P \rightarrow B$, and also the following autoepistemic axiom which represents the belief that not knowing B ''now'' implies its negation.

Axiom 1 $(\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B]$[3]

This axiom actually says that if I didn't know *at the previous step* that I have an older brother, then conclude that I have no older brother. The reason for this delay in time is discussed below in Section 6.3.

The scenarios illustrate the following behaviors:

- If B is among the wffs of which the agent is aware at step i, but not one that is believed at step i, then the agent will come to know this fact ($\neg K(i, B)$, that it was not believed at step i) at step $i+1$. As a consequence of this, other information may be deduced. In this case, the agent concludes $\neg B$ from the autoepistemic axiom (Axiom 1). Clearly the Now predicate plays a critical role. Section 6.2.1 below illustrates this case.
- The agent must refrain from such negative introspection when in fact B is already known; see Section 6.2.2.
- A conflict may occur if something is coming to be known while negative introspection is simultaneously leading to its negation. The third illustration (see Section 6.2.3 below) shows this being resolved in an intuitive manner (though not one that will generalize as much as we would like; see Section 8.2 for a discussion of this).

[1]This problem is used although, according to Moore, it technically does not involve ''true'' default reasoning. In Chapter 8 we discuss a standard simple default about birds typically flying.

[2]P might be something like ''My parents have two sons,'' together with appropriate axioms.

[3]It appears that some arithmetic is involved here, although our reasoner has no arithmetic capabilities. Simple syntactic devices can be used to avoid this difficulty. For instance, $K(i-1, \alpha)$ can be replaced by $J(i, \alpha)$, with the intuitive meaning that α was known ''just a moment ago'', i.e., at $i-1$. Alternatively, we can use successor notation for natural numbers.

6.2 Three Scenarios

Each scenario presented is a synopsis of actual computer-generated results (the code for which can be found in Appendix D). Refer to page 20 for the inference-function INF_B.

6.2.1 Simple negative introspection succeeds

In this example the agent is not able to deduce the proposition B, that he has an older brother, and hence is able to deduce $\neg B$, that he does *not* have an older brother. See Figure 6.1. Here, and in Figures 6.2 and 6.3, for ease of reading we underline in each step those wffs which are new (i.e., which appear through other than inheritance). For the purposes of illustration, let i be arbitrary, but fixed, and let

$$OBS_{B_1}(j) = \begin{cases} \{P \to B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B]\} & \text{if } j = i \\ \emptyset & \text{otherwise} \end{cases}$$

Since B is not an i-observation (and thus is not an i-theorem), the agent uses Rule 5, the negative introspection rule, to conclude $\neg K(i, B)$ at step $i + 1$. At step $i + 2$ the agent concludes $\neg B$ from the autoepistemic knowledge stated above (Axiom 1) and the use of the extended version of modus ponens, Rule 4.

$$\begin{array}{rl}
i: & \underline{Now(i)}, \underline{P \to B}, \underline{(\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B]} \\[1em]
i+1: & \underline{Now(i+1)}, P \to B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B], \underline{\neg K(i, B)}, \\
 & \underline{\neg K(i, \neg B)}, \underline{\neg K(i, P)} \\[1em]
i+2: & \underline{Now(i+2)}, P \to B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B], \neg K(i, B), \\
 & \neg K(i, \neg B), \neg K(i, P), \underline{\neg B}, \underline{\neg K(i+1, B)}, \underline{\neg K(i+1, \neg B)}, \underline{\neg K(i+1, P)}
\end{array}$$

Figure 6.1: Negative introspection succeeds

6.2.2 Simple negative introspection fails (appropriately)

In this example, let

$$OBS_{B_2}(j) = \begin{cases} \{P \to B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B], B\} & \text{if } j = i \\ \emptyset & \text{otherwise} \end{cases}$$

Thus the agent has B at step i, and is blocked (appropriately for this example) from deducing the wffs $\neg K(i, B)$ and $\neg B$. See Figure 6.2.

$$\begin{array}{rl}
i: & \underline{Now(i)}, \underline{P \to B}, \underline{(\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B]}, \underline{B} \\[1em]
i+1: & \underline{Now(i+1)}, P \to B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \to \neg B], B, \underline{\neg K(i, \neg B)}, \\
 & \underline{\neg K(i, P)}
\end{array}$$

Figure 6.2: Negative introspection fails appropriately

Note that a traditional final-tray-like approach could produce quite similar behavior to that seen in Figures 6.1 and 6.2 if it is endowed with a suitable introspection device, although it would not have the real-time step-like character we are trying to achieve. A traditional final-tray-like approach would have severe difficulties with this next example, however.

6.2.3 Introspection contradicts other deduction

In this example, let

$$OBS_{B_3}(j) = \begin{cases} \{P \rightarrow B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B], P\} & \text{if } j = i \\ \emptyset & \text{otherwise} \end{cases}$$

In Figure 6.3 we see then that the agent does not have B at step i, but is able to *deduce* B at step $i+1$ from $P \rightarrow B$ and P at step i. Since the agent is *aware* (in our sense) of B at step i, and yet does not have B as a *conclusion* at i, it will deduce $\neg K(i, B)$ at step $i+1$. Thus both B and $\neg K(i, B)$ are concluded at step $i+1$. At step $i+2$ Axiom 1, together with $Now(i+1)$ and $\neg K(i, B)$ and Rule 4, will produce $\neg B$. A conflict results, which is noted at step $i+3$. This then inhibits inheritance of both B and $\neg B$ to step $i+4$. Although neither B nor $\neg B$ is *inherited* to step $i+4$, B is *re-deduced* at step $i+4$ via *modus ponens*. Thus B ''wins out'' over $\neg B$ due to its existing justification in other wffs, while $\neg B$'s justification is ''too old'': $\neg K(i+2, B)$, rather than $\neg K(i, B)$, would be needed. We see then that the conflict resolves due to the special nature of the time-bound ''now'' feature of introspection.

$$\begin{array}{rl}
i: & \underline{Now(i)}, \underline{P \rightarrow B}, \underline{(\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B]}, \underline{P} \\[1em]
i+1: & \underline{Now(i+1)}, P \rightarrow B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B], P, \underline{B}, \underline{\neg K(i, B)}, \\
 & \underline{\neg K(i, \neg B)} \\[1em]
i+2: & \underline{Now(i+2)}, P \rightarrow B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B], P, B, \neg K(i, B), \\
 & \neg K(i, \neg B), \underline{\neg B}, \underline{\neg K(i+1, \neg B)} \\[1em]
i+3: & \underline{Now(i+3)}, P \rightarrow B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B], P, B, \neg K(i, B), \\
 & \neg K(i, \neg B), \neg B, \neg K(i+1, \neg B), \underline{Contra(i+2, B, \neg B)} \\[1em]
i+4: & \underline{Now(i+4)}, P \rightarrow B, (\forall x)[(Now(x) \wedge \neg K(x-1, B)) \rightarrow \neg B], P, \neg K(i, B), \\
 & \neg K(i, \neg B), \neg K(i+1, \neg B), Contra(i+2, B, \neg B), \underline{B}, \underline{Contra(i+3, B, \neg B)}
\end{array}$$

Figure 6.3: Introspection conflicts with other deduction and resolves

A traditional final-tray-like approach would encounter difficulties with this third example, for at step $i+2$ there is a contradiction. This means that the final tray for a tray-like model of a reasoning agent would simply be filled with all wffs in the language---and no basis for a resolution possible *within* such a logic.

6.3 Discussion

We see then that $SL_7(OBS_{B_i}, INF_B)$ is a promising theory for handling simple default cases involving negative introspection such as we have encountered with the *Brother problem*. When it's appropriate to make the default conclusion (that no older brother exists), the conclusion is indeed inferred. When circumstances should block the default conclusion (when in fact it *is* known that an older brother exists), the conclusion is not inferred. In the third case where self-reflection allows the default inference *at the same time* that another inference leads to the conclusion that an older brother does truly exist, a contradiction results. The logic allows the contradiction to be sorted out; after only a couple of steps the situation stabilizes, with B being believed, and $\neg B$ not being believed.

It is worth noting that, although this simple method of contradiction-handling produces the correct results in this particular scenario, it is due to the fact that one of the contradictory beliefs was inferred through the use of a default rule involving negative introspection. It may be, for example, that a direct contradiction arises due to a series of inferences, none of which involves negative introspection. In this case, although neither belief would be inherited to the next step (once the contradiction was noted), *both* would possibly be re-deduced in the next step.

Typicalities (such as birds typically flying) can lead to such difficulties. Complex interacting defaults must be sorted out and contradiction resolution is not so easily done. A more sophisticated method of contradiction-handling is necessary. If the bird example, however, is formulated in a similar fashion to the *Brother problem*, then the simple contradiction-handling mechanism presented here works well. The details and complexities of this distinction, using the bird example for illustration, are discussed in Chapter 8.

The curious reader may be puzzled by the particular formulation of Axiom 1 that was used in this problem. (See page 23.) Why was it necessary for the reasoner to introspect on the *previous* step, instead of the *current* one? This is a general phenomenon of temporal constraint that pervades our development. Consider the process of concluding by default, on the basis of not knowing X "now," that X is false (where X is any assertion, possibly dependent on time). This requires the agent to determine at time i that it does not know X at time i. Intuitively, certain beliefs have accumulated at time i, and it is only *after* these beliefs are formed (say at time $i+1$) that the new belief, that X is not among the former, can be formed. If this conclusion were to be drawn at time i instead, then we would not be dealing with a fixed set of beliefs for time i. We, in fact, would have a two-stage production in which beliefs are gathered initially and then an introspective process is allowed to add to that set. But this leads to severe ambiguities, for the very process of inserting, say, $\neg K(i, X)$ into the beliefs at time i results in something new being known, something that was *not* really known at time i, namely $\neg K(i, X)$ itself.

Now suppose we grant that some oracle manages to place all negative introspective conclusions about the time-i belief set *into that very same set*. This unfortunately forces an infinite set of beliefs into that set, since there are infinitely many unknown formulas at any given step. Our approach of real-time reasoning requires a finite belief set at all steps. Thus this approach cannot be taken, and we must forego the luxury of having the agent be able to know that it doesn't know a given fact *now*; instead the best that can be done is to know that it didn't know the fact a moment ago, when it last was able to scan its belief set. Thus the agent's self-knowledge must lag slightly behind. Hence the formulation of Axiom 1 which was given.

We now present a theorem about the consistency of the three step-logics which were defined in this chapter. All three step-logics share the same inference-function, INF_B (defined on page 20). The observation-functions, $OBS_{B_1}, \ldots, OBS_{B_3}$, are as defined in Sections 6.2.1 thru 6.2.3.

Theorem 6.1 *The following are true about the consistency of each of the SL_7 theories given in the brother examples:*

1. *$SL_7(OBS_{B_1}, INF_B)$ is step-wise consistent.*
2. *$SL_7(OBS_{B_2}, INF_B)$ is step-wise consistent.*
3. *$SL_7(OBS_{B_3}, INF_B)$ is eventually consistent (but not step-wise consistent[4]).*

Proof: We briefly sketch the proof of part 1 of this theorem. Parts 2 and 3 are similar; part 3 involves constructing a model for each step after the last inconsistent step (step $i+3$).

Since $OBS_{B_1}(j) = \emptyset$, for $j < i$, by Remark 5.5 (page 21), if $j < i$, $\alpha \in \vdash_j$ iff $\alpha = Now(j)$. Therefore every step in $SL_7(OBS_{B_1}, INF_B)$ up to and including step $i-1$ is consistent. From step i on we have additional theorems which must be considered, due to the fact that $OBS_{B_1}(i)$ is not empty. Note that $OBS_{B_1}(i)$ itself is consistent.

To show that step i and all subsequent steps are consistent, we propose a model M_j for each step j. In each M_j interpret the predicates in the following way: $K \equiv false$, $B \equiv false$, $P \equiv false$, $Now(k) \equiv k = j$, where P is any predicate other than K, B, or Now. We can then see that we have a model for each of steps i thru $i+2$. Noting that for an arbitrary step $i+k$, $k > 2$,

[4]This is why a traditional final-tray-like approach would encounter difficulties with this example.

$$\vdash_{i+k} = \left\{ \begin{array}{l} Now(i+k), \\ P \to B, \\ (\forall x)[(Now(x) \land \neg K(x-1, B)) \to \neg B], \\ \neg B, \\ \neg K(i, \neg B), \neg K(i+1, \neg B), \\ \neg K(i, B), \ldots, \neg K(i+k-1, B), \\ \neg K(i, P), \ldots, \neg K(i+k-1, P) \end{array} \right\}$$

we see that, again, M_{i+k} is an appropriate model. Therefore, by Theorem 3.16 (page 14), $SL_7(OBS_{B_1}, INF_B)$ is step-wise consistent. $\Box$

6.4 Implementation of $SL_7(OBS_{B_3}, INF_B)$

The code for the implementation of the *Brother problem* can be found in Appendix D.

6.5 Summary

In this chapter we demonstrated that $SL_7(OBS_{B_i}, INF_B)$ is a suitable theory for handling a simple default involving negative introspection. Three separate SL_7-theories were presented. Each was suitable for modelling a particular scenario involving the *Brother problem*. The three scenarios differed in the information that was given to each agent. Therefore, although each step-logic shared the same inference-function INF_B, they differed in the definition of the observation-function. The three scenarios demonstrated the following:

- When it was appropriate to make a default conclusion (that no older brother exists), the conclusion was inferred.
- When circumstances should block the default conclusion, the conclusion was not inferred.
- When the default conclusion was inferred *at the same time* that another inference led to the conclusion that an older brother existed, a contradiction resulted which was subsequently resolved in favor of the conclusion that an older brother exists.

7 The Wise-men Problem(s)

7.1 Statement of the Problem

We reiterate the problem which was described in Section 2.3. A king wishes to know whether his three advisors are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down. The wise man in the back (wise man #3) can see the backs of the other two men. The man in the middle (wise man #2) can only see the one wise man in front of him (wise man #1); and the wise man in front (wise man #1) can see neither wise man #3 nor wise man #2. The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the color of his own card. Each wise man must announce the color of his own card as soon as he knows what it is. (The first to correctly announce the color of his own card will be aptly rewarded.) All know that this will happen. The room is silent; then, after several minutes, wise man #1 says ''My card is white!''.

We assume in this puzzle that the wise men do not lie, that they all have the same reasoning capabilities, and that they can all think at the same speed. We then can postulate that the following reasoning took place. Each wise man knows there is at least one white card. If the cards of wise man #2 and wise man #1 were black, then wise man #3 would have been able to announce immediately that his card was white. They all realize this (they are all truly wise). Since wise man #3 kept silent, either wise man #2's card is white, or wise man #1's is. At this point wise man #2 would be able to determine, if wise man #1's were black, that his card was white. They all realize this. Since wise man #2 also remains silent, wise man #1 knows his card must be white.

In Section 7.2 we describe and propose a model for the version of this problem in which there are only two men. Once the Two-wise-men version was tackled, it was easier to propose a model of the problem in which there are three men. This is done in Section 7.3.

7.2 The Two-wise-men Problem

7.2.1 Statement of the Problem

In this puzzle the king has just two wise men and two cards, at least one of which is white. Wise man #2 sits behind wise man #1, so wise man #1 can see nothing, and wise man #2 can see wise man #1's card. Wise man #2 is unable to identify the color of his card. Wise man #1 is then able to determine that his card must be white.

The reasoning involved in this version of the puzzle is much simpler than in the three-wise-men version. Wise man #2 can see the color of wise man #1's card. If it were black, then wise man #2 would know, since there is at least one white card, that his card was white. Wise man #1 knows this. Wise man #2 says nothing. Therefore, wise man #1's card must not be black, but rather white.

The step-logic used to model this problem is defined in Figure 7.1 below. The problem is modelled from wise man #1's point of view. The observation-function contains all the axioms that wise man #1 needs to solve the problem, and the inference-function provides the allowable rules of inference. The language is that defined in Section 5.2 with the following additions:

- $K_j(i, x)$, instead of $K(i, x)$ is used. $K_j(i, x)$ is intended to mean ''wise man j knows x at step i.''
- $U(i, x)$ expresses the fact that an utterance of x is made at step i.
- $s(i)$ is the successor function (where $s^k(0)$ is used as an abbreviation for $\underbrace{s(s(\cdots(s}_{k}(0))\cdots)))$.

OBS is defined as follows.

$$OBS(i) = \begin{cases} \left\{ \begin{array}{l} (\forall i)(\forall x)(\forall y)[K_2(i, x \rightarrow y) \rightarrow (K_2(i, x) \rightarrow K_2(s(i), y))] \\ K_2(s(0), B_1 \rightarrow W_2) \\ (B_1 \rightarrow K_2(s(0), B_1)) \\ (\neg B_1 \rightarrow W_1) \\ (\forall i)[\neg U(s(i), W_2) \rightarrow \neg K_2(i, W_2)] \\ (\forall i)[\neg K_1(s(i), U(i, W_2)) \rightarrow \neg U(i, W_2)] \end{array} \right\} & \text{if } i = 1 \\ \\ \emptyset & \text{otherwise} \end{cases}$$

The inference rules given here correspond to an inference-function, INF_{W_2}. For any given history, INF_{W_2} returns the set of all immediate consequences of Rules 1--6 applied to the last step in that history. Note that Rule 5 is the only default rule.

Rule 1 : $\dfrac{i : \ldots}{i+1 : \ldots, \alpha}$ if $\alpha \in OBS(i+1)$

Rule 2 : $\dfrac{i : \ldots, \alpha, (\alpha \rightarrow \beta)}{i+1 : \ldots, \beta}$ Modus ponens

Rule 3 : $\dfrac{i : \ldots, P_1\overline{a}, \ldots, P_n\overline{a}, (\forall \overline{x})[(P_1\overline{x} \wedge \ldots \wedge P_n\overline{x}) \rightarrow Q\overline{x}]}{i+1 : \ldots, Q\overline{a}}$ Extended modus ponens

Rule 4 : $\dfrac{i : \ldots, \neg\beta, (\alpha \rightarrow \beta)}{i+1 : \ldots, \neg\alpha}$

Rule 5 : $\dfrac{i : \ldots}{i+1 : \ldots, \neg K_1(s^i(0), U(s^{i-1}(0), W_2))}$ if $U(s^{i-1}(0), W_2) \notin \vdash_i$, $i > 1$

Rule 6 : $\dfrac{i : \ldots, \alpha}{i+1 : \ldots, \alpha}$ Inheritance

Figure 7.1: OBS_{W_2} and INF_{W_2} for the Two-wise-men Problem

- W_i and B_i express the facts that i's card is white, and i's card is black, respectively.

The axioms of $OBS_{W_2}(1)$ have the following intuitive meaning. (Refer to Figure 7.1.) Wise man #1 knows the following:

1. Wise man #2 uses the rule of *modus ponens*.
2. Wise man #2 knows at step $s(0)$ that if my card is black, then his is white.
3. If my card is black, then wise man #2 knows this at step $s(0)$.
4. If my card is not black, then it is white.
5. If there is no utterance of W_2 at a given time-step, then wise man #2 didn't know that his card was white (i.e. W_2) at the previous time-step.
6. If I don't know at a given time-step that there has been an utterance of W_2, then there was no utterance of W_2 at the previous time-step. (I would know that an utterance of W_2 was made one step after it is uttered.)

The rules of inference are the same as those for the *Brother problem* (see Figure 5.1 on page 20), with the following exceptions:

- The rule that infers $Now(i)$ is not needed.
- The rule that recognizes contradictions is not needed.
- The introspective rule has been replaced with a more specific introspective rule. (Wise man #1 can only introspect on what utterances have been made, rather than on all sub-formulas of which he is aware.)
- The rule of inheritance is more general: *everything* is inherited from one step to the next.
- The rule for extended *modus ponens* allows an arbitrary number of variables.
- Rule 4 is new.

7.2.2 Solution

The solution to the problem is given in Figures 7.2 and 7.3 below. The step number is listed on the left. The reason (rule used) for each inference is listed on the right. Only those wffs which are of interest are shown. (There are many wffs involving introspection which are left out of the figure for purposes of clarity.)

In step 1, we see that all the initial axioms ($OBS_{W_2}(1)$) have been inferred through the use of Rule 1. In step 2, (a) has been deduced through the use of Rule 3. (a) says that if wise man #2 knows B_1 at step $s(0)$, then wise man #2 will know W_2 at step $s(s(0))$. The rest of the wffs listed in step 2 have been inferred through Rule 6, the rule of inheritance. At step 5, wise man #1 negatively introspects to determine that no utterance of W_2 was made at step 3. Note the time delay: wise man #1 is able to prove *at step 5* that he did not know *at step 4* of an utterance made *at step 3*. This phenomenon is discussed in the previous chapter, in Section 6.3. At step 6, wise man #1 can then conclude that indeed no utterance of W_2 was made at step 3. The reasoning continues from step to step, and in step 10, wise man #1 is finally able to prove that his card is white.

0:		$\emptyset$	
1:	(a)	$(\forall \boldsymbol{i})(\forall \boldsymbol{x})(\forall \boldsymbol{y})[K_2(\boldsymbol{i}, \boldsymbol{x} \to \boldsymbol{y}) \to (K_2(\boldsymbol{i}, \boldsymbol{x}) \to K_2(\boldsymbol{s}(\boldsymbol{i}), \boldsymbol{y}))]$	(R1)
	(b)	$K_2(\boldsymbol{s}(0), B_1 \to W_2)$	(R1)
	(c)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R1)
	(d)	$(\neg B_1 \to W_1)$	(R1)
	(e)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R1)
	(f)	$(\forall \boldsymbol{i})[\neg K_1(\boldsymbol{s}(\boldsymbol{i}), U(\boldsymbol{i}, W_2)) \to \neg U(\boldsymbol{i}, W_2)]$	(R1)
2:	(a)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R3,1a,1b)
	(b)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,1c)
	(c)	$(\neg B_1 \to W_1)$	(R6,1d)
	(d)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R6,1e)
	(e)	$(\forall \boldsymbol{i})[\neg K_1(\boldsymbol{s}(\boldsymbol{i}), U(\boldsymbol{i}, W_2)) \to \neg U(\boldsymbol{i}, W_2)]$	(R6,1f)
3:	(a)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R6,2a)
	(b)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,2b)
	(c)	$(\neg B_1 \to W_1)$	(R6,2c)
	(d)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R6,2d)
	(e)	$(\forall \boldsymbol{i})[\neg K_1(\boldsymbol{s}(\boldsymbol{i}), U(\boldsymbol{i}, W_2)) \to \neg U(\boldsymbol{i}, W_2)]$	(R6,2e)
4:	(a)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R6,3a)
	(b)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,3b)
	(c)	$(\neg B_1 \to W_1)$	(R6,3c)
	(d)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R6,3d)
	(e)	$(\forall \boldsymbol{i})[\neg K_1(\boldsymbol{s}(\boldsymbol{i}), U(\boldsymbol{i}, W_2)) \to \neg U(\boldsymbol{i}, W_2)]$	(R6,3e)

Figure 7.2: Solution to the Two-wise-men Problem---Part I

5:	(a)	$\neg K_1(\boldsymbol{s}^4(0), U(\boldsymbol{s}^3(0), W_2))$	(R5)
	(b)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R6,4a)
	(c)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,4b)
	(d)	$(\neg B_1 \to W_1)$	(R6,4c)
	(e)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R6,4d)
	(f)	$(\forall \boldsymbol{i})[\neg K_1(\boldsymbol{s}(\boldsymbol{i}), U(\boldsymbol{i}, W_2)) \to \neg U(\boldsymbol{i}, W_2)]$	(R6,4e)
6:	(a)	$\neg U(\boldsymbol{s}^3(0), W_2)$	(R3,5a,5f)
	(b)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R6,5b)
	(c)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,5c)
	(d)	$(\neg B_1 \to W_1)$	(R6,5d)
	(e)	$(\forall \boldsymbol{i})[\neg U(\boldsymbol{s}(\boldsymbol{i}), W_2) \to \neg K_2(\boldsymbol{i}, W_2)]$	(R6,5e)
7:	(a)	$\neg K_2(\boldsymbol{s}^2(0), W_2)$	(R3,6a,6e)
	(b)	$(K_2(\boldsymbol{s}(0), B_1) \to K_2(\boldsymbol{s}^2(0), W_2))$	(R6,6b)
	(c)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,6c)
	(d)	$(\neg B_1 \to W_1)$	(R6,6d)
8:	(a)	$\neg K_2(\boldsymbol{s}(0), B_1)$	(R4,7a,7b)
	(b)	$(B_1 \to K_2(\boldsymbol{s}(0), B_1))$	(R6,7c)
	(c)	$(\neg B_1 \to W_1)$	(R6,7d)
9:	(a)	$\neg B_1$	(R4,8a,8b)
	(b)	$(\neg B_1 \to W_1)$	(R6,8c)
10:		W_1	(R2,9a,9b)

Figure 7.3: Solution to the Two-wise-men Problem---Part II

7.2.3 Discussion

It may seem that the axioms with which wise man #1 is endowed are a bit contrived. Admittedly, perhaps they are. I feel, however, that what should come across is the fact that step-logic has been a useful vehicle for formulating and solving this problem in which the time that something occurs is important. Wise man #1 does indeed determine that ‘‘if wise man #2 knew the color of his card, he would have announced it by now.’’ (See steps 6 and 7.) Wise man #1 then reasons backwards from here to determine that his card must not be black (step 9), and hence must be white (step 10).

It is interesting to note that wise man #1 needs to know very little about wise man #2 and how he reasons. We could, for instance, have given wise man #1 the information that wise man #2 is just as clever as he, and thus has all the same rules of inference. This much knowledge was not necessary, however. The only rule of wise man #2’s that wise man #1 needed to know about was that of *modus ponens*.

Wise man #1 needed to know three additional facts about wise man #2: 1) that wise man #2 could see the color of wise man #1’s card; 2) that wise man #2 knew there was at least one white card; and 3) that wise man #2 would announce the color of his card as soon as he knew it. Let’s consider each of these in turn.

1. Wise man #2 could see the color of wise man #1’s card. The third axiom of $OBS_{W_2}(1)$ represents this fact. It actually states that if wise man #1’s card is black, then wise man #2 would know this. The analogous axiom for the situation where wise man #1’s card is white was not necessary for the problem. To keep things symmetric, it could have been included, but I felt it was unnecessary since it was not needed in the solution.

2. Wise man #2 knows there is at least one white card. The second axiom in $OBS_{W_2}(1)$ is meant to represent this. It doesn’t quite say this, however; it says instead that wise man #2 knows that if wise man #1’s card is black, then his is white. The reader may feel that this perhaps short-circuits the problem---wise man #2 should be able to determine this on his own based on an axiom such as $K_2(s(0), W_1 \vee W_2)$. Wise man #2 has no tautologies (nor does wise man #1 for that matter) and hence it was much easier to simply endow wise man #2 with this axiom directly. I felt this distinction was not crucial to the problem. See Section 7.3.3 for a discussion of the difficulty with tautologies.

3. Wise man #2 would announce the color of his card as soon as he knew it. The fifth axiom of $OBS_{W_2}(1)$ is meant to represent this fact. Again, as in point 1, only half of this fact is represented. Wise man #1 does not need to know what wise man #2 would do in the case that wise man #2’s card is black; it is sufficient for wise man #1 to know that if wise man #2 knew his card was white, he would say so immediately (that is, in the step after he has determined that his card is indeed white).

Note that wise man #1 needs to know only that wise man #2 knows points 1 and 2 *at step 1*. Wise man #1 doesn’t need to know that wise man #2 may know these two facts later in time as well. We will see that this is *not* sufficient in the *Three-wise-men problem*.

It should also be pointed out that the step-logic described here, which I have labeled $SL_7(OBS_{W_2}, INF_{W_2})$, is not strictly an SL_7 logic! That is to say that, although time and self-knowledge are critical components of wise man #1’s reasoning, retraction is not used by wise man #1. (The rule of inheritance forces *all* wffs to be inherited from one step to the next.) This logic should technically be labeled $SL_5(OBS_{W_2}, INF_{W_2})$. Since it seemed that it was only this *particular* formulation that didn’t happen to require retraction (because wise man #1 can reason backwards instead of forwards), I chose to retain the SL_7 title.

7.2.4 Implementation of $SL_7(OBS_{W_2}, INF_{W_2})$

$SL_7(OBS_{W_2}, INF_{W_2})$ has been implemented on an IBM PC-AT using Arity Prolog. An example run can be found in Appendix C. The code used to implement this solution can be found in Appendix D.

7.3 The Three-wise-men Problem

7.3.1 Statement of the Problem

Now that the *Two-wise-men problem* has been modelled, the mechanics of that problem can be brought to bear on the three-wise-men version. The *Three-wise-men problem* is described on page 28. Recall that we are assuming that each wise man announces the color of his card as soon as he knows what it is. All know that this will happen. Therefore, if wise man #2, for instance, is able to determine that wise man #3 would have been able to figure out by now that wise man #3's card is white, and wise man #2 has heard nothing, then wise man #2 knows that wise man #3 does *not* know the color of his card. We therefore do not have to know when exactly 30 minutes has gone by (as stated in the original problem). The step-logic used to model this problem is defined in Figures 7.4 and 7.5 below.

OBS_{W_3} is defined as follows.

$$OBS_{W_3}(i) = \begin{cases} \left\{ \begin{array}{l} (\forall j)K_2(j, (\forall i)(\forall x)(\forall y)[K_3(i, x \rightarrow y) \rightarrow \\ \qquad\qquad\qquad\qquad (K_3(i, x) \rightarrow K_3(s(i), y))]) \\ (\forall j)K_2(j, K_3(s(0), (B_1 \wedge B_2) \rightarrow W_3)) \\ (\forall j)K_2(j, (B_1 \wedge B_2) \rightarrow K_3(s(0), B_1 \wedge B_2)) \\ (\forall j)K_2(j, \neg(B_1 \wedge B_2) \rightarrow (B_1 \rightarrow W_2)) \\ (\forall j)K_2(j, (\forall i)[\neg U(s(i), W_3) \rightarrow \neg K_3(i, W_3)]) \\ (\forall i)(\forall x)[\neg K_1(s(i), U(i, x)) \rightarrow \neg U(i, x)] \\ (\forall i)[\neg U(i, W_3) \rightarrow K_2(s(i), \neg U(i, W_3))] \\ (\forall i)(\forall x)(\forall y)[K_2(i, x \rightarrow y) \rightarrow (K_2(i, x) \rightarrow K_2(s(i), y))] \\ (\forall i)(\forall x)(\forall x')(\forall y)(\forall y') \\ \quad [(K_2(i, \neg(x \wedge x') \rightarrow (y \wedge y')) \wedge K_2(i, \neg(x \wedge x'))) \rightarrow \\ \quad\; K_2(s(i), y \wedge y')] \\ (\forall j)(\forall k)(\forall z)(\forall z')(\forall w) \\ \quad [(K_2(j, (\forall i)(\forall x)(\forall y)[K_3(i, x \rightarrow y) \rightarrow \\ \qquad\qquad\qquad\qquad (K_3(i, x) \rightarrow K_3(s(i), y))]) \wedge \\ \quad\; K_2(j, K_3(k, (z \wedge z') \rightarrow w))) \rightarrow \\ \quad K_2(s(j), K_3(k, z \wedge z') \rightarrow K_3(s(k), w))] \\ (\forall j)(\forall k) \\ \quad [(K_2(j, (\forall i)[\neg U(s(i), W_3) \rightarrow \neg K_3(i, W_3)]) \wedge \\ \quad\; K_2(j, \neg U(s(k), W_3))) \rightarrow \\ \quad K_2(s(j), \neg K_3(k, W_3))] \\ (\forall i)(\forall x)(\forall y)[(K_2(i, x \rightarrow y) \wedge K_2(i, \neg y)) \rightarrow K_2(s(i), \neg x)] \\ (\forall i)(\forall x)(\forall x')(\forall y) \\ \quad [(K_2(i, (x \wedge x') \rightarrow y) \wedge K_2(i, \neg y)) \rightarrow K_2(s(i), \neg(x \wedge x'))] \\ (\forall i)[B_1 \rightarrow K_2(i, B_1)] \\ (\neg B_1 \rightarrow W_1) \\ (\forall i)[\neg U(s(i), W_2) \rightarrow \neg K_2(i, W_2)] \end{array} \right\} & \text{if } i = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

Figure 7.4: OBS_{W_3} for the Three-wise-men Problem

As in the two-wise-man version, the problem is modelled from wise man #1's point of view. The observation-function contains all the axioms that wise man #1 needs to solve the problem, and the inference-function provides the allowable rules of inference. The language is the same as that for the *Two-wise-men problem* (see page 28).

The axioms of $OBS_{W_3}(1)$ have the following intuitive meaning. (Refer to Figure 7.4.) Wise man #1 knows the following:

1. Wise man #2 always knows (i.e. knows at every time-step) that wise man #3 uses the rule of *modus ponens*.

INF_{W_3} is the same as INF_{W_2} (see Figure 7.1, page 29) augmented with the following additional rules (where Rule 8 replaces Rule 5).

$$\textbf{Rule 7}: \quad \frac{i : \ldots, \neg Q\overline{a}, (\forall \overline{x})(P\overline{x} \rightarrow Q\overline{x})}{i+1 : \ldots, \neg P\overline{a}}$$

$$\textbf{Rule 8}: \quad \frac{i : \ldots}{i+1 : \ldots, \neg K_1(s^i(0), U(s^{i-1}(0), W_j))} \quad \text{if } U(s^{i-1}(0), W_j) \notin \vdash_i, \; j = 2, 3, i > 1$$

$$\textbf{Rule 9}: \quad \frac{i : \ldots, (\forall j)K_2(j, \alpha)}{i+1 : \ldots, K_2(s^i(0), \alpha)}$$

Figure 7.5: INF_{W_3} for the Three-wise-men Problem

2. Wise man #2 always knows that wise man #3 knows at time-step $s(0)$ that if both my card and wise man #2's card are black, then wise man #3's card is white.
3. Wise man #2 always knows that if both my card and his card are black, wise man #3 would know this fact at time-step $s(0)$.
4. Wise man #2 always knows that if it's not the case that both my card and his are black, then if mine is black, then his is white.[1]
5. Wise man #2 always knows that if there's no utterance of W_3 at a given time-step, then wise man #3 did not know W_3 at the previous step. (Wise man #2 always knows that there will be an utterance of W_3 the step after wise man #3 has proved that his card is white.)
6. If I don't know about a given utterance, then it has not been made at the previous step.
7. If there's no utterance of W_3 at a given time-step, then wise man #2 will know this at the next step.
8. Wise man #2 uses the rule of *modus ponens*.
9. Same meaning as previous axiom. (This is necessary since wise man #1 and wise man #2 reason within first-order logic.)
10. Wise man #2 uses the extended rule of *modus ponens*.
11. Same meaning as previous axiom. (Again necessary since wise man #1 and wise man #2 reason within first-order logic.)
12. Wise man #2 uses the contra-positive of *modus ponens*.
13. Same meaning as previous axiom. (Again necessary since wise man #1 and wise man #2 reason within first-order logic.)
14. If my card is black, then wise man #2 always knows this.
15. If my card is not black, then it is white.
16. If there is no utterance of W_2 at a given time-step, then wise man #2 doesn't know at the previous step that his card is white. (There would be an utterance of W_2 the step after wise man #2 knows his card is white.)

Note that the first five axioms are analogous to those needed in the two-wise-men version, where wise man #1 is reasoning about wise man #2's reasoning about wise man #3, instead of wise man #1 reasoning about wise man #2. Also note that all the axioms but one that were needed in the two-wise-men version are

[1]In other words, if wise man #2 knows that at least one of our cards is white, then my card being black would mean that his is white. Indeed, this axiom gives wise man #2 quite a bit of information, perhaps too much. (He should be able to deduce some of this himself.) This is discussed in more detail in Section 7.3.3.

also needed here (although two of these are in slightly generalized form). The second axiom of $OBS_{W_2}(1)$ is not part of $OBS_{W_3}(1)$, but is deduced by wise man #1 in a subsequent step. See Section 7.3.2 below.

The rules of inference are the same as those for the *Two-wise-men problem* (see Figure 7.1 on page 29), with the following exceptions:

- The introspective rule allows wise man #1 to introspect on utterances of W_3 as well as W_2.
- An extended version of Rule 4 was added (Rule 7).
- Rule 9, a rule of instantiation, was added.

7.3.2 Solution

The solution to the problem is given in Figures 7.6 and 7.7 below. The step number is listed on the left. The reason (rule used) for each inference is listed on the right. To allow for ease of reading, only the wffs in which we are interested are shown at each step. In addition, none of the inherited wffs are shown. This means that a rule appears to be operating on a step other than the previous one, but the wffs involved have, in fact, actually been inherited to the appropriate step.

0:		$\emptyset$	
1:	(a)	$(\forall j)K_2(j, (\forall i)(\forall x)(\forall y)[K_3(i, x \to y) \to (K_3(i, x) \to K_3(s(i), y))])$	(R1)
	(b)	$(\forall j)K_2(j, K_3(s(0), (B_1 \wedge B_2) \to W_3))$	(R1)
	(c)	$(\forall j)K_2(j, (B_1 \wedge B_2) \to K_3(s(0), B_1 \wedge B_2))$	(R1)
	(d)	$(\forall j)K_2(j, \neg(B_1 \wedge B_2) \to (B_1 \to W_2))$	(R1)
	(e)	$(\forall j)K_2(j, (\forall i)[\neg U(s(i), W_3) \to \neg K_3(i, W_3)])$	(R1)
	(f)	$(\forall i)(\forall x)[\neg K_1(s(i), U(i, x)) \to \neg U(i, x)]$	(R1)
	(g)	$(\forall i)[\neg U(i, W_3) \to K_2(s(i), \neg U(i, W_3))]$	(R1)
	(h)	$(\forall i)(\forall x)(\forall y)[K_2(i, x \to y) \to (K_2(i, x) \to K_2(s(i), y))]$	(R1)
	(h′)	$(\forall i)(\forall x)(\forall x')(\forall y)(\forall y')$ $[(K_2(i, \neg(x \wedge x') \to (y \wedge y')) \wedge K_2(i, \neg(x \wedge x'))) \to K_2(s(i), y \wedge y')]$	(R1)
	(i)	$(\forall j)(\forall k)(\forall z)(\forall z')(\forall w)$ $[(K_2(j, (\forall i)(\forall x)(\forall y)[K_3(i, x \to y) \to (K_3(i, x) \to K_3(s(i), y))]) \wedge K_2(j, K_3(k, (z \wedge z') \to w))) \to K_2(s(j), K_3(k, z \wedge z') \to K_3(s(k), w))]$	(R1)
	(i′)	$(\forall j)(\forall k)$ $[(K_2(j, (\forall i)[\neg U(s(i), W_3) \to \neg K_3(i, W_3)]) \wedge K_2(j, \neg U(s(k), W_3))) \to K_2(s(j), \neg K_3(k, W_3))]$	(R1)
	(j)	$(\forall i)(\forall x)(\forall y)[(K_2(i, x \to y) \wedge K_2(i, \neg y)) \to K_2(s(i), \neg x)]$	(R1)
	(j′)	$(\forall i)(\forall x)(\forall x')(\forall y)$ $[(K_2(i, (x \wedge x') \to y) \wedge K_2(i, \neg y)) \to K_2(s(i), \neg(x \wedge x'))]$	(R1)
	(k)	$(\forall i)[B_1 \to K_2(i, B_1)]$	(R1)
	(l)	$(\neg B_1 \to W_1)$	(R1)
	(m)	$(\forall i)[\neg U(s(i), W_2) \to \neg K_2(i, W_2)]$	(R1)
⋮			
5:	(a)	$\neg K_1(s^4(0), U(s^3(0), W_3))$	(R8)
	(b)	$K_2(s^4(0), (\forall i)(\forall x)(\forall y)$ $[K_3(i, x \to y) \to (K_3(i, x) \to K_3(s(i), y))])$	(R9,1a)
	(c)	$K_2(s^4(0), K_3(s(0), (B_1 \wedge B_2) \to W_3))$	(R9,1b)
	(d)	$K_2(s^4(0), (\forall i)[\neg U(s(i), W_3) \to \neg K_3(i, W_3)])$	(R9,1e)

Figure 7.6: Solution to the Three-wise-men Problem---Part I

6:	(a)	$\neg U(\boldsymbol{s}^3(0), W_3)$	(R3,5a,1f)
	(b)	$K_2(\boldsymbol{s}^5(0), K_3(\boldsymbol{s}(0), B_1 \wedge B_2) \rightarrow K_3(\boldsymbol{s}^2(0), W_3))$	(R3,5b,5c,1i)
7:	(a)	$K_2(\boldsymbol{s}^4(0), \neg U(\boldsymbol{s}^3(0), W_3))$	(R3,6a,1g)
	(b)	$K_2(\boldsymbol{s}^6(0), (B_1 \wedge B_2) \rightarrow K_3(\boldsymbol{s}(0), B_1 \wedge B_2))$	(R9,1c)
8:	(a)	$K_2(\boldsymbol{s}^5(0), \neg K_3(\boldsymbol{s}^2(0), W_3))$	(R3,7a,5d,1i′)
	(b)	$K_2(\boldsymbol{s}^7(0), \neg(B_1 \wedge B_2) \rightarrow (B_1 \rightarrow W_2))$	(R9,1d)
9:		$K_2(\boldsymbol{s}^6(0), \neg K_3(\boldsymbol{s}(0), B_1 \wedge B_2))$	(R3,8a,6b,1j)
10:		$K_2(\boldsymbol{s}^7(0), \neg(B_1 \wedge B_2))$	(R3,9,7b,1j′)
11:		$K_2(\boldsymbol{s}^8(0), B_1 \rightarrow W_2)$	(R3,10,8b,1h′)
12:	(a)	$(K_2(\boldsymbol{s}^8(0), B_1) \rightarrow K_2(\boldsymbol{s}^9(0), W_2))$	(R3,11,1h)
	(b)	$\neg K_1(\boldsymbol{s}^{11}(0), U(\boldsymbol{s}^{10}(0), W_2))$	(R8)
13:		$\neg U(\boldsymbol{s}^{10}(0), W_2)$	(R3,12b,1f)
14:		$\neg K_2(\boldsymbol{s}^9(0), W_2)$	(R3,13,1m)
15:		$\neg K_2(\boldsymbol{s}^8(0), B_1)$	(R4,14,12a)
16:		$\neg B_1$	(R7,15,1k)
17:		W_1	(R2,16,1l)

Figure 7.7: Solution to the Three-wise-men Problem---Part II

In step 1 we see that all the initial axioms ($OBS_{W_3}(1)$) have been inferred through the use of Rule 1. Nothing of interest is inferred in steps 2 through 4. In step 5, wise man #1 is able to negatively introspect and determine that no utterance of W_3 was made at step 3. Note the time delay. The rest of the wffs shown in step 5 were all inferred through the use of Rule 9, the rule of instantiation. Wise man #1 needs to know that wise man #2 knows these particular facts at step 4. The reasoning continues from step to step. Note that at step 11, wise man #1 has been able to deduce that wise man #2 knows that if wise man #1's card is black, then his is white. This is the axiom of $OBS_{W_2}(1)$ that is *not* in $OBS_{W_3}(1)$. From this step on, we essentially have the *Two-wise-men problem*. In step 17 wise man #1 is finally able to deduce that his card is white.

7.3.3 Discussion

Several points of contrast can be drawn between this version and the two-wise-men version.

1. Unlike the two-wise-men version, where wise man #1 needed only to know about a *single* rule of inference used by wise man #2, in this version wise man #1 needs to know *several* rules used by wise man #2: *modus ponens*, extended *modus ponens*, and the contra-positive of *modus ponens*. Because wise man #1 reasons within first-order logic, these three rules required the use of six axioms.
2. In the three-wise-men version, unlike in the two-wise-men version, it is *not* sufficient for wise man #1 to know that wise man #2 has certain beliefs *at step 1*. Wise man #1 must know that wise man #2 *always* holds these beliefs.
3. What wise man #2 needs to know about wise man #3 is analogous to what wise man #1 needed to know about wise man #2 in the two-wise-men version. So, for instance, wise man #2 must know that wise man #3 uses the rule of *modus ponens* (and this is the only rule of wise man #3's about which wise man #2 must know). Also wise man #2 needs only to know that wise man #3 has certain beliefs *at step 1*.

Again, as in the two-wise-men version, we are faced with the fact that the axioms aren't very general. In the two-wise-men version we could perhaps get away with it. Here it seems a great deal of work is taken from the wise men and put into a few simple axioms. The problem is wise man #1 has no tautologies available to him.

Adding tautologies provides for several complications. If we were to simply endow the agent with all tautologies (through, for instance, the three axiom schemata used in SL^0; see Chapter 4), the agent would then have an infinite number of beliefs at each step. As we have previously discussed, this is something we do *not* want. The agent can negatively introspect exactly *because* he has only a finite number of beliefs at each step. We are not willing to give this up. We could ''feed in'' tautologies the way we did in SL_0 (again see Chapter 4), but this would give us an exceptionally large number of beliefs at each step. Although this would work, the idea is to keep the number of beliefs at any given step to a minimum. Feeding in tautologies, together with some mechanism for keeping the belief-set small, would be a very reasonable solution. (See the discussion on relevance in Chapter 8.)

Another possible solution to the tautology problem is the use of resolution. If all wffs were put into clause form, and then, instead of *modus ponens*, the agent were to use the *resolution principle*, it seems that tautologies may not be needed. This idea, although quite promising, has not been investigated in any detail by the author.

Many formulations of the *Three-wise-men problem* have involved the use of common knowledge or common belief (see [Kon84] and [KL87] in particular). For instance, a possible axiom might be $C(W_1 \vee W_2 \vee W_3)$: it is common knowledge that at least one card is white. I found adding the common knowledge concept introduced unnecessary complications. This was to a large degree due to the fact that I have modelled the reasoning *from wise man #1's point of view*, rather than using a meta-language that describes the reasoning of all three (as [Kon84, KL87] have both done). I feel that this is more in the spirit of step-logics, where the idea is to allow the reasoner itself enough power (with no outside ''oracle'' intervention) to solve the problem. Thus we model the agent directly, rather than using a meta-theory as a model.

7.3.4 Implementation of $SL_7(OBS_{W_3}, INF_{W_3})$

$SL_7(OBS_{W_2}, INF_{W_2})$ has been implemented on an IBM PC-AT using Arity Prolog. The example took 3 $\frac{3}{4}$ hours to run on an IBM PC-AT. (It took about 15 minutes on a VAX 11-785.) At step 17,the step in which the wise man #1 is finally able to prove that his spot is white, there are a total of 259 wffs proven.[2]

An example run can be found in Appendix C. The code used to implement this solution can be found in Appendix D.

[2]This could be drastically reduced if we were to employ some sort of relevance mechanism. See Chapter 8 for a discussion of this.

7.4 Summary

In this chapter we defined $SL_7(OBS_{W_2}, INF_{W_2})$ and $SL_7(OBS_{W_3}, INF_{W_3})$ and showed that they were suitable theories for modelling the *Two-wise-men* and the *Three-wise-men* problems, respectively. The *Two-wise-men problem* was tackled to help us understand what was needed to solve the *Three-wise-men problem*. Both problems were modelled from wise man #1's point of view. OBS_{W_2} (OBS_{W_3}) represents the information that wise man #1 needs to know in order to solve the problem.

8 Summary, Discussion and Future Work

This chapter summarizes the work in this thesis and discusses several weaknesses with the current formulation of step-logics. Several areas for further research are suggested.

8.1 Summary

The formulation of step-logics demonstrates that a formal treatment of commonsense reasoning situated in time not only is possible, but can remain largely deductive in nature. Negative introspection becomes computationally tractable, while forcing a time delay between knowledge and self-knowledge. See Section 6.3 for a discussion of why this time delay is necessary.

Unlike traditional logic, step-logic can tolerate contradictions. A chief drawback of more traditional logics is what we call the *swamping problem*: from a contradiction all wffs come to be theorems.[1] In step-logic, however, it is not at all critical whether a contradiction is *instantly* resolved. Inferences are drawn step-by-step, and hence the possible spread of invalid conclusions based on a contradiction also occurs only step-by-step. This makes it possible to effectively control the spread. This is illustrated with a simple example of default reasoning (see Chapter 6).

We have formalized a form of introspection relevant to default reasoning, which occurs in real-time. The introspective rule introduced demonstrates that it is possible to have real-time self-reference affecting the very course of deduction itself.

We have presented both formal solutions and computer implementations of the *Brother problem* and the *Three-wise-men problem* (see Chapters 6 and 7, respectively).

8.2 Discussion

We have illustrated that step-logic provides a formalism in which contradictions can arise and be subsequently resolved. With the current mechanisms, however, only very special types of contradictions can be resolved. We saw in the *Brother problem* (Chapter 6) that the reasoner could infer a belief α based on the fact that he did not believe its negation. Later, however, through a sequence of inferences, he may come to believe $\neg\alpha$. The contradiction would be noted, causing neither α nor $\neg\alpha$ to be inherited to the next step. The only way α or $\neg\alpha$ would again enter into his belief set would be through a deduction involving any rule other than the rule of inheritance. In the *Brother problem* we saw that the justification for B still existed, so the agent was able to re-deduce B. $\neg B$, on the other hand, which was inferred based on the lack of B, could no longer be deduced. Thus the contradiction was resolved.

This simple mechanism of merely not inheriting the two contradictory beliefs works well in this situation. But what happens when we are dealing with typicalities, such as the fact that birds typically fly? If we formulate this problem using the following axiom:

$$(\forall x)(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x),$$

then we have an analogous situation to the *Brother problem*, and all works well. See Figure 8.1 for an example. In step 3 both $flies(T)$ and $\neg flies(T)$ have been deduced, but by step 5 the contradiction has

[1]This is not a problem in *implementations* of standard logics, however, since the theorems of the logic are not all drawn instantaneously.

Let the inference-function be INF_B (see Figure 5.1, page 20), and let

$$OBS(j) = \begin{cases} \left\{ \begin{array}{l} penguin(T), \\ (\forall x)(penguin(x) \rightarrow bird(x)), \\ (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\ (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \\ \rightarrow flies(x)] \end{array} \right\} & \text{if } j = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

We then have the following sequence of deductions:

0 : $\emptyset$

1 : $Now(1), penguin(T),$
$(\forall x)(penguin(x) \rightarrow bird(x)),$
$(\forall x)(penguin(x) \rightarrow \neg flies(x)),$
$(\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$

2 : $\ldots, Now(2), penguin(T),$
$(\forall x)(penguin(x) \rightarrow bird(x)),$
$(\forall x)(penguin(x) \rightarrow \neg flies(x)),$
$(\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x)],$
$\neg K(1, \neg flies(T)), bird(T), \neg flies(T)$

3 : $\ldots, Now(3), penguin(T),$
$(\forall x)(penguin(x) \rightarrow bird(x)),$
$(\forall x)(penguin(x) \rightarrow \neg flies(x)),$
$(\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x)],$
$\neg K(1, \neg flies(T)), bird(T), \neg flies(T), flies(T)$

4 : $\ldots, Now(4), penguin(T),$
$(\forall x)(penguin(x) \rightarrow bird(x)),$
$(\forall x)(penguin(x) \rightarrow \neg flies(x)),$
$(\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x)],$
$\neg K(1, \neg flies(T)), bird(T), \neg flies(T), flies(T), contra(3, flies(T), \neg flies(T))$

5 : $\ldots, Now(5), penguin(T),$
$(\forall x)(penguin(x) \rightarrow bird(x)),$
$(\forall x)(penguin(x) \rightarrow \neg flies(x)),$
$(\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \rightarrow flies(x)],$
$\neg K(1, \neg flies(T)), bird(T), contra(3, flies(T), \neg flies(T)),$
$contra(4, flies(T), \neg flies(T)), \neg flies(T)$

Note that at step 5 $\neg flies(T)$ has been re-deduced, but $flies(T)$ has not.

Figure 8.1: Bird example in which contradiction handling works

been resolved in favor of $\neg flies(T)$; that is, $flies(T)$ is no longer a theorem, but $\neg flies(T)$ is. If, however, we wish instead to use the simpler axiom,

$$(\forall x)(bird(x) \rightarrow flies(x)),$$

the contradiction handling mechanism allows the contradiction to persist. See Figure 8.2. Obviously a more

Let the inference-function be INF_B (see Figure 5.1, page 20), and let

$$OBS(i) = \begin{cases} \left\{ \begin{array}{l} penguin(T), \\ (\forall x)(penguin(x) \rightarrow bird(x)), \\ (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\ (\forall x)(bird(x) \rightarrow flies(x)) \end{array} \right\} & \text{if } i = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

We then have the following sequence of deductions:

$$\begin{array}{ll}
0: & \emptyset \\
\\
1: & Now(1), penguin(T), \\
& (\forall x)(penguin(x) \rightarrow bird(x)), \\
& (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\
& (\forall x)(bird(x) \rightarrow flies(x)) \\
\\
2: & \ldots, Now(2), penguin(T), \\
& (\forall x)(penguin(x) \rightarrow bird(x)), \\
& (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\
& (\forall x)(bird(x) \rightarrow flies(x)) \\
& bird(T), \neg flies(T) \\
\\
3: & \ldots, Now(3), penguin(T), \\
& (\forall x)(penguin(x) \rightarrow bird(x)), \\
& (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\
& (\forall x)(bird(x) \rightarrow flies(x)) \\
& bird(T), \neg flies(T), flies(T) \\
\\
4: & \ldots, Now(4), penguin(T), \\
& (\forall x)(penguin(x) \rightarrow bird(x)), \\
& (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\
& (\forall x)(bird(x) \rightarrow flies(x)) \\
& bird(T), \neg flies(T), flies(T), contra(3, flies(T), \neg flies(T)) \\
\\
5: & \ldots, Now(5), penguin(T), \\
& (\forall x)(penguin(x) \rightarrow bird(x)), \\
& (\forall x)(penguin(x) \rightarrow \neg flies(x)), \\
& (\forall x)(bird(x) \rightarrow flies(x)) \\
& bird(T), contra(3, flies(T), \neg flies(T)), contra(4, flies(T), \neg flies(T)), \\
& \neg flies(T), flies(T)
\end{array}$$

Note that at step 5 *both* $\neg flies(T)$ and $flies(T)$ have been re-deduced.

Figure 8.2: Bird example in which contradiction handling does not work

sophisticated contradiction handling mechanism is needed.

This formulation of the bird problem points out a very important problem with the current method for handling contradictions. Suppose (as in the bird problem) that B and $\neg B$ are deducible from other beliefs

$P_1, \ldots, P_n$ (without the use of the introspective rule, so that earlier steps contain indirect contradictions). It is then not enough to merely block the inheritance of B and $\neg B$; rather the roots of the contradiction, $P_1, \ldots, P_n$, must be investigated in order to unwind the contradiction.

Related to this is the problem of not inheriting the consequents of a contradictory belief. Although the swamping problem is much less serious in step-logic than in final-tray-like logics (since invalid conclusions spread only a step at a time), the problem is not completely eliminated. Since the agent *can* continue reasoning step-by-step even if a contradiction exists, new conclusions based on the contradictory beliefs may be made. These new (invalid) conclusions may then be the basis for further deductions. It is this proliferation of invalid conclusions that needs to be curtailed. Currently the contradictory beliefs themselves are the only beliefs which are retracted. A mechanism similar to that of [Doy82] and [deK86a], put into a real-time framework, may be appropriate for handling both the consequents and the antecedents of the contradictory beliefs.

One of the difficulties we encountered in our efforts to represent real-time reasoning involved the concept of ''now.'' The approach we found most useful for the *Brother problem* is the one given in Rules 1 and 7, coupled with Rules 3 and 4 for *modus ponens* (see Figure 5.1 on page 20), where Now is a predicate symbol with special treatment regarding inheritance. There are, however, variations on this example where this treatment is not completely satisfactory. In particular, a difficulty can arise when there is a detachment of a ''$Now(j)$'' sub-formula from the rest of the formula, producing what we call a ''dangling time-parameter.''

As an example, consider what would happen in the scenario in Figure 6.3 (page 25) if, instead of Axiom 1 (page 23), we had used the following:

Axiom 2 $(\forall x)[Now(x) \rightarrow (\neg K(x-1, B) \rightarrow \neg B)]$,

We would then get an intermediate conclusion at step $i+2$, namely,

$$\neg K(i, B) \rightarrow \neg B. \tag{8.1}$$

The problem is that (8.1) inherits to future steps, even though the intended significance of i in (8.1) was that it was the *current* time-step (i.e., linked to $Now(i)$) rather than any particular fixed step; by step $i+2$, the term i has lost its tie to the wff $Now(i)$, and so ''dangles'' inappropriately. *Modus ponens* can then be used with (8.1) to conclude $\neg B$ at any step after $i+2$ in which $\neg K(i, B)$ appears. Since we do have $\neg K(i, B)$ at step $i+1$ and all subsequent steps (inherited via Rule 7), the conclusion $\neg B$ is re-deduced from step $i+3$ on, despite the contradiction resolution mechanism.

This emphasizes that B's merely not being known some time ago is an insufficient reason to conclude $\neg B$. That is, if we have deduced $\neg B$ from $\neg K(i, B)$ at step $i+2$, but later (or in the meantime) we conclude B, we no longer want to be able to deduce $\neg B$. Any satisfactory treatment, then, should refer to the fact that the agent does not know B at the *current* time step, before autoepistemically deducing $\neg B$.

The particular formulation of the *Brother problem* that we presented in Chapter 6 satisfies this condition due to the special form of the autoepistemic axiom (Axiom 1). A similar, but even safer, approach is that of employing a special purpose *inference rule* (instead of the autoepistemic *axiom*) such as:

$$\frac{Now(i) \wedge \neg K(i-1, B)}{\neg B} \tag{8.2}$$

However, we prefer a treatment that allows the agent to explicitly represent such a train of deduction, as in Axiom 1, for then the agent also has the possibility of reasoning about this very process of reasoning. The fact that the alternate version (Axiom 2) above is not satisfactory suggests that dangling time-parameters be avoided in a more general (less syntax-dependent) way.

8.3 Future Directions

There is still much to be done with step-logics. In Section 8.2 we discussed the need for more general contradiction-handling mechanisms. We also pointed out the need to treat dangling time-parameters in a less syntax-dependent way. There are many other areas for further research.

As mentionned earlier, we would like to make broad use of a retraction mechanism to keep the number of beliefs at any given step to a reasonable size. One of the crucial features of step-logic is the finiteness of the belief-set at any given step. But when this set becomes overwhelmingly large (which seems to happen rather quickly in the step-logics we have investigated), although it is still finite, it is no longer a cognitively plausible model.

We can think of retraction in terms of lack of inheritance. Introducing a notion of relevance along the lines pursued in [DMP86a], where a computational model of memory was proposed, would help in this regard. We envision a given step as being a sort of short-term memory (STM) as in [DMP86a], where only a very small number of beliefs are held at any given time. This would be the working set of beliefs. This requires the use of a notion of relevance: those beliefs relevant to the current task at hand would be retained; all others would not. This of course would require a mechanism for retrieving beliefs which later *do* become relevant---it would no longer be sufficient to merely inherit beliefs from one step to the next. Step-logics as defined are sufficiently general to allow for this sort of change.

Having such a mechanism in place would help to make the *Three-wise-men* model more plausible: tautologies could be fed in, with those that are relevant being inherited, and those that are not being retracted. With the introduction of tautologies we could then make our three-wise-men axioms more general, and hence more intuitively appealing.

The current formulation of step-logic is not sufficient to handle the *Little Nell* problem discussed in Section 2.3. Although the theory allows the agent to recognize that its reasoning takes time, there currently is no mechanism in place to estimate how long it will take to solve a particular problem. This involves reasoning about actions and plans. Suitable axioms need to be developed to handle this.

Any new theory is better understood with an appropriate associated semantics. In Section 3.2 we defined the notion of a step-model, and with it were able to prove several important theorems: every step-logic is sound with respect to step-models; and a step-logic that has an associated step-model is step-wise consistent (at each step, the set of beliefs is consistent). We see then that our notion of step-model was sufficient to show that certain step-logics are consistent. But, since step-logics are supposed to be able to model the reasoning of a *fallible* agent, we find that many interesting step-logics are in fact *inconsistent*. The current formulation of step-models must be generalized to provide an appropriate semantics for these more interesting inconsistent theories.

It appears that any step-logic appropriate for broad commonsense reasoning should be self-stabilizing, where, after a period of time, the steps would become and remain consistent. The idea is that, although the step-logic is meant to model a *fallible* agent, the agent should not be continually on the trail of emerging contradictions. Of course in the case of a particularly unfriendly environment which has an infinite supply of new contradictions to current beliefs, the logic would not self-stabilize.

Also interesting to consider is the following. If the *course* of reasoning can be altered (by retraction, etc.), then why not also the *rules* of reasoning? That is, learning based on past experiences ought to allow the system to change its own set of rules, so that over time it could become more adapted to its environment. This would then come that much closer to formalizing the ‘‘computer individual’’ postulated by Nilsson (see [Nil83]), where the system would have a constantly changing model of the world, learning and benefiting from its experiences.

A Proofs of Theorems about $\mathbf{SL^0}$

This appendix contains the proofs of the results found in Section 4.2.2. The major result is that of *analytic completeness* in SL^0, i.e., for any given time i, and for any given wff α in a supposed reasoning agent's language, SL^0 can prove either that the agent knows (has proved or otherwise determined) α at i, or that the agent has not done so. Thus the agent's reasoning over time is completely characterized, in the sense that we know what it has and has not established at each moment.

A.1 The Workings

The reasoning agent's formalism is denoted by SL_0. The symbols of SL_0 are $\neg$, $\rightarrow$, and the propositional letters, $P_1, P_2, P_3 \ldots$ The set of wffs, $\mathcal{L}(SL_0)$, is defined recursively as follows,

1. All propositional letters are wffs.
2. If α and β are wffs, then so are $(\neg\alpha)$ and $(\alpha \rightarrow \beta)$.
3. An expression is a wff only if it can be built up using (1) and/or (2).

The axioms of SL_0 are fed in step-by-step, and the only rule of inference is *modus ponens*. We continue our convention of using Greek letters to stand for wffs in the agent's language.

SL^0 (i.e., the meta-theory) models such a propositional reasoning agent. To help differentiate SL^0 and SL_0, we use ‘‘implies’’ and ‘‘not’’ as function symbols of SL^0 to designate implication and negation, respectively, of the agent's wffs. SL^0 is a first order theory with equality, with the rules of *modus ponens* and *generalization*. Frequent use is made of the *Deduction Theorem*. We follow the treatment of [Men87].

A.2 Axioms of SL^0

The axioms are those listed in Section 4.2.1.

A.3 Analytic Completeness

A.3.1 Preliminary Lemmas

Before presenting the proof of analytic completeness, we first give the following lemmas and their proofs.

Lemma A.1 (Newborn Lemma) $SL^0 \vdash (\forall\alpha)(\neg K(0,\alpha))$.
The agent initially can prove nothing.

Proof: By THM,
$SL^0 \vdash (\forall\alpha)[K(0,\alpha) \rightarrow [Feed(\alpha,0) \vee (\exists j)(\exists k)(\exists\beta)(K(j,\beta) \wedge K(k, implies(\beta,\alpha)) \wedge j < 0 \wedge k < 0)]]$.
By TABULARASA, $SL^0 \vdash (\forall\alpha)\neg(\exists j)(\exists k)(\exists\beta)(K(j,\beta) \wedge K(k, implies(\beta,\alpha)) \wedge j < 0 \wedge k < 0)$.
Therefore, $SL^0 \vdash (\forall\alpha)[K(0,\alpha) \rightarrow Feed(\alpha,0)]$.
And, by ALP, $SL^0 \vdash (\forall\alpha)[K(0,\alpha) \rightarrow [Ax(\alpha) \wedge l(\alpha) \leq 0 \wedge p(\alpha) < 0]]$.
Now, by P4, $SL^0 \vdash (\forall\alpha)\neg(p(\alpha) < 0)$.
Hence, $SL^0 \vdash (\forall\alpha)\neg[Ax(\alpha) \wedge l(\alpha) \leq 0 \wedge p(\alpha) < 0]$.
And we conclude, $SL^0 \vdash (\forall\alpha)(\neg K(0,\alpha))$. □

Lemma A.2 (Monotonicity Lemma) $SL^0 \vdash (\forall i)(\forall \alpha)[K(i, \alpha) \rightarrow K(i+1, \alpha)]$.
Once a belief is held, it remains.

Proof: Let $\mathcal{T} = SL^0 + \{K(i, \alpha)\}$.

Then, by THM, ALP, and MP,
$\mathcal{T} \vdash (Ax(\alpha) \wedge l(\alpha) \leq i \wedge p(\alpha) < i) \vee$
$(\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < i \wedge k < i)$.

Now, $\mathcal{T} \vdash i < (i+1)$ [AXIOM: $SL^0 \vdash i < j$, for $i < j, i, j \in \mathcal{N}$].
Also, $\mathcal{T} \vdash (\forall k)(k < i \rightarrow k < (i+1))$, for any $i \in \mathcal{N}$ [axiom of SL^0].

Thus,
$\mathcal{T} \vdash [\ (Ax(\alpha) \wedge l(\alpha) \leq i \wedge p(\alpha) < i) \vee$
$(\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < i \wedge k < i)] \rightarrow$
$[(Ax(\alpha) \wedge l(\alpha) \leq (i+1) \wedge p(\alpha) < (i+1)) \vee$
$(\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < (i+1) \wedge k < (i+1))]$.

Then, by *modus ponens*,
$\mathcal{T} \vdash (Ax(\alpha) \wedge l(\alpha) \leq (i+1) \wedge p(\alpha) < (i+1)) \vee$
$(\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < (i+1) \wedge k < (i+1))$.

By THM, ALP, and MP, we then have, $\mathcal{T} \vdash K(i+1, \alpha)$.
Then, by the *Deduction theorem*, $SL^0 \vdash K(i, \alpha) \rightarrow K(i+1, \alpha)$.
And, by *generalization*, $SL^0 \vdash (\forall i)(\forall \alpha)[K(i, \alpha) \rightarrow K(i+1, \alpha)]$. □

Lemma A.3 $SL^0 \vdash (\forall \alpha)(\neg K(1, \alpha))$.
The agent knows nothing at time step 1.

Proof: By THM and MP,
$SL^0 \vdash (\forall \alpha)[K(1, \alpha) \rightarrow [Feed(\alpha, 1) \vee (\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < 1 \wedge k < 1)]]$.
But by Lemma A.1 and by TABULARASA,
$SL^0 \vdash (\forall \alpha)\neg(\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < 1 \wedge k < 1)$.
Hence, $SL^0 \vdash (\forall \alpha)[K(1, \alpha) \rightarrow Feed(\alpha, 1)]$.
And, by ALP, $SL^0 \vdash (\forall \alpha)[K(1, \alpha) \rightarrow [Ax(\alpha) \wedge l(\alpha) \leq 1 \wedge p(\alpha) < 1]]$.
Now, by Lemma A.4 (below), $SL^0 \vdash (\forall \alpha)\neg[Ax(\alpha) \wedge l(\alpha) \leq 1]$.
Hence, $SL^0 \vdash (\forall \alpha)\neg[Ax(\alpha) \wedge l(\alpha) \leq 1 \wedge p(\alpha) < 1]$.
Therefore, $SL^0 \vdash (\forall \alpha)(\neg K(1, \alpha))$. □

Lemma A.4 $SL^0 \vdash (\forall \alpha)[Ax(\alpha) \rightarrow l(\alpha) \geq 2]$.

Proof: Let $\mathcal{T} = SL^0 + \{Ax(\alpha)\}$.
Then, by Lemma A.5 (below), $\mathcal{T} \vdash (\exists \beta)(\exists \gamma)(\exists \delta)(\alpha = implies(\beta, implies(\gamma, \delta)))$.

Now, since,
$SL^0 \vdash (\forall \alpha)(\forall \beta)(\forall \gamma)(\forall \delta)[\ \alpha = implies(\beta, implies(\gamma, \delta)) \longleftrightarrow$
$(\exists \varepsilon)[\alpha = implies(\beta, \varepsilon) \wedge \varepsilon = implies(\gamma, \delta)]]$,

we have, $\mathcal{T} \vdash (\exists \beta)(\exists \gamma)(\exists \delta)(\exists \varepsilon)[\alpha = implies(\beta, \varepsilon) \wedge \varepsilon = implies(\gamma, \delta)]$.
Now, by LN3, $\mathcal{T} \vdash (\forall \alpha)(\forall \beta)(\forall \gamma)[\alpha = implies(\beta, \gamma) \rightarrow l(\alpha) = l(\beta) + l(\gamma) + 1]$.
Hence, $\mathcal{T} \vdash (\exists \beta)(\exists \gamma)(\exists \delta)(\exists \varepsilon)[l(\alpha) = l(\beta) + l(\varepsilon) + 1) \wedge l(\varepsilon) = l(\gamma) + l(\delta) + 1]$.
That is, $\mathcal{T} \vdash (\exists \beta)(\exists \gamma)(\exists \delta)(\exists \varepsilon)[l(\alpha) = l(\beta) + l(\gamma) + l(\delta) + 2]$.
Then, by LN4, $\mathcal{T} \vdash l(\alpha) \geq 2$.
Hence, by the *Deduction theorem*, $SL^0 \vdash Ax(\alpha) \rightarrow l(\alpha) \geq 2$.
And by *generalization*, we have, $SL^0 \vdash (\forall \alpha)[Ax(\alpha) \rightarrow l(\alpha) \geq 2]$. □

Lemma A.5 $SL^0 \vdash (\forall\alpha)[Ax(\alpha) \rightarrow (\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(\beta, implies(\gamma, \delta)))]$.

Proof:

By AX, $SL^0 \vdash (\forall\alpha)[Ax(\alpha) \longleftrightarrow$

$$\begin{aligned}
&[(\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)))\vee\\
&(\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)),\\
&\qquad implies(implies(not(\gamma), \beta), \gamma))\vee\\
&(\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma, \delta)),\\
&\qquad implies(implies(\beta, \gamma),\\
&\qquad\quad implies(\beta, \delta)))]].
\end{aligned}$$

Now, since, $SL^0 \vdash (\forall\alpha)[(\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta))) \longleftrightarrow$
$(\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(\beta, implies(\gamma, \delta)) \wedge \beta = \delta)]$

and $SL^0 \vdash (\forall\alpha)[(\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)),$
$implies(implies(not(\gamma), \beta), \gamma))) \longleftrightarrow$

$$\begin{aligned}
&(\exists\beta)(\exists\gamma)(\exists\delta)(\exists\varepsilon)(\alpha = implies(\delta, implies(\varepsilon, \gamma))\wedge\\
&\qquad \delta = implies(not(\gamma), not(\beta))\wedge\\
&\qquad \varepsilon = implies(not(\gamma), \beta))]
\end{aligned}$$

and $SL^0 \vdash (\forall\alpha)[(\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma, \delta)),$
$implies(implies(\beta, \gamma),$
$implies(\beta, \delta))) \longleftrightarrow$

$$\begin{aligned}
&(\exists\beta)(\exists\gamma)(\exists\delta)(\exists\varepsilon)(\exists\zeta)(\exists\eta)(\alpha = implies(\varepsilon, implies(\zeta, \eta))\wedge\\
&\qquad \varepsilon = implies(\beta, implies(\gamma, \delta))\wedge\\
&\qquad \zeta = implies(\beta, \gamma)\wedge\\
&\qquad \eta = implies(\beta, \delta))],
\end{aligned}$$

we have, $SL^0 \vdash (\forall\alpha)[Ax(\alpha) \longleftrightarrow$

$$\begin{aligned}
&[(\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(\beta, implies(\gamma, \delta)) \wedge \beta = \delta)\vee\\
&(\exists\beta)(\exists\gamma)(\exists\delta)(\exists\varepsilon)(\alpha = implies(\delta, implies(\varepsilon, \gamma))\wedge\\
&\qquad \delta = implies(not(\gamma), not(\beta))\wedge\\
&\qquad \varepsilon = implies(not(\gamma), \beta))\vee\\
&(\exists\beta)(\exists\gamma)(\exists\delta)(\exists\varepsilon)(\exists\zeta)(\exists\eta)\\
&\qquad (\alpha = implies(\varepsilon, implies(\zeta, \eta))\wedge\\
&\qquad \varepsilon = implies(\beta, implies(\gamma, \delta))\wedge\\
&\qquad \zeta = implies(\beta, \gamma)\wedge\\
&\qquad \eta = implies(\beta, \delta))]].
\end{aligned}$$

Thus, $SL^0 \vdash (\forall\alpha)[Ax(\alpha) \rightarrow (\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(\beta, implies(\gamma, \delta)))]$. $\square$

Lemma A.6 (Inequality Lemma) *If α and β are distinct wffs $\in \mathcal{L}(SL_0)$, then* $SL^0 \vdash \alpha \neq \beta$.

Proof: By strong induction on the number of connectives in α.

Base case: Let α and β be distinct wffs, and let the number of connectives in α be 0, i.e., α is a propositional letter.

Suppose α has index i (i.e., $\alpha = P_i$).

Case a: $\beta = P_j$, for some $j \in \mathcal{N}, j \neq i$.

By EQ1, $SL^0 \vdash \alpha \neq \beta$.

Case b: $\beta = not(\gamma)$, for some $\gamma \in \mathcal{L}(SL_0)$.

By EQ3, $SL^0 \vdash \alpha \neq \beta$.

Case c: $\beta = implies(\gamma, \delta)$, for some $\gamma, \delta \in \mathcal{L}(SL_0)$.

By EQ2, $SL^0 \vdash \alpha \neq \beta$.

Assume $SL^0 \vdash \alpha \neq \beta$, for all distinct $\alpha, \beta \in \mathcal{L}(SL_0)$, where α has fewer than k connectives, $k > 0$.
To show true for α with k connectives.
Case 1: $\alpha = not(\gamma)$, for some $\gamma \in \mathcal{L}(SL_0)$.

Case 1a: $\beta = P_i$, for some $i \in \mathcal{N}$.

By EQ3, $SL^0 \vdash \alpha \neq \beta$.

Case 1b: $\beta = not(\delta)$, for some $\delta \in \mathcal{L}(SL_0)$.

Note that γ and δ must be distinct.
Since α has k connectives, γ must have $k - 1$ connectives.
Hence, by the inductive hypothesis, $SL^0 \vdash \gamma \neq \delta$.
Then, by EQ6, $SL^0 \vdash not(\gamma) \neq not(\delta)$.
That is, $SL^0 \vdash \alpha \neq \beta$.

Case 1c: $\beta = implies(\delta, \varepsilon)$, for some $\delta, \varepsilon \in \mathcal{L}(SL_0)$.

By EQ4, $SL^0 \vdash \alpha \neq \beta$.

Case 2: $\alpha = implies(\gamma, \delta)$, for some $\gamma, \delta \in \mathcal{L}(SL_0)$.

Case 2a: $\beta = P_i$, for some $i \in \mathcal{N}$.

By EQ2, $SL^0 \vdash \alpha \neq \beta$.

Case 2b: $\beta = not(\varepsilon)$, for some $\varepsilon \in \mathcal{L}(SL_0)$.

By EQ4, $SL^0 \vdash \alpha \neq \beta$.

Case 2c: $\beta = implies(\varepsilon, \zeta)$, for some $\varepsilon, \zeta \in \mathcal{L}(SL_0)$.

Note that either $\gamma \neq \varepsilon$ or $\delta \neq \zeta$.
Since α has k connectives, γ and δ each must have fewer than k connectives.
Hence, by the inductive hypothesis, $SL^0 \vdash \gamma \neq \varepsilon$ or $SL^0 \vdash \delta \neq \zeta$.
In either case we then have, by EQ5, $SL^0 \vdash implies(\gamma, \delta) \neq implies(\varepsilon, \zeta)$.
That is, $SL^0 \vdash \alpha \neq \beta$.

Therefore, for all distinct $\alpha, \beta \in \mathcal{L}(SL_0)$, $SL^0 \vdash \alpha \neq \beta$. $\square$

Lemma A.7 (Boundedness Lemma) *Let $i \in \mathcal{N}$, $i \geq 2$. Then $\exists \alpha_1 \ldots \alpha_{n_i} \in \mathcal{L}(SL_0)$, such that*

$$SL^0 \vdash (\forall \alpha)[K(i, \alpha) \longleftrightarrow (\alpha = \alpha_1 \vee \ldots \vee \alpha = \alpha_{n_i})].$$

Proof: By strong induction on i.
Base case: $i = 2$.

By THM, $SL^0 \vdash (\forall \alpha)(K(2, \alpha) \longleftrightarrow [Feed(\alpha, 2) \vee Mp(\alpha, 2)])$.

Then, by MP, we have,
$SL^0 \vdash (\forall \alpha)(K(2, \alpha) \longleftrightarrow$
$[Feed(\alpha, 2) \vee (\exists j)(\exists k)(\exists \beta)$
$(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < 2 \wedge k < 2)])$.

Now, by Lemma A.1, Lemma A.3, and TABULARASA,
$SL^0 \vdash (\forall \alpha) \neg (\exists j)(\exists k)(\exists \beta)(K(j, \beta) \wedge K(k, implies(\beta, \alpha)) \wedge j < 2 \wedge k < 2)$.
Hence, $SL^0 \vdash (\forall \alpha)(K(2, \alpha) \longleftrightarrow Feed(\alpha, 2))$.

Then, by Lemma A.8 (below), we have,

$$\begin{aligned} SL^0 \vdash (\forall\alpha)(K(2,\alpha) \longleftrightarrow [&\alpha = implies(P_0, implies(P_0, P_0)) \vee \\ &\alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \\ &\alpha = implies(P_1, implies(P_1, P_1))]). \end{aligned}$$

Assume true for $i < k$, $k > 2$.
To show true for $i = k$, i.e. $\exists\alpha_1 \ldots \alpha_{n_k} \in \mathcal{L}(SL_0)$, such that
$SL^0 \vdash (\forall\alpha)[K(k,\alpha) \longleftrightarrow (\alpha = \alpha_1 \vee \ldots \vee \alpha = \alpha_{n_k})]$.
By MP,
$SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \longleftrightarrow (\exists j)(\exists l)(\exists\beta)(K(j,\beta) \wedge K(l, implies(\beta,\alpha)) \wedge j < k \wedge l < k)]$.
By Lemma A.2, this gives us,
$SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \longleftrightarrow (\exists\beta)(K(k-1,\beta) \wedge K(k-1, implies(\beta,\alpha)))]$.

Then, by the inductive hypothesis,

$$\begin{aligned} SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \longleftrightarrow (\exists\beta)[&(\beta = \beta_1 \vee \ldots \vee \beta = \beta_{n_{k-1}}) \wedge \\ &(implies(\beta,\alpha) = \beta_1 \vee \ldots \vee \\ &\ implies(\beta,\alpha) = \beta_{n_{k-1}})]]. \end{aligned}$$

But this is true iff,

$$\begin{aligned} &SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \longleftrightarrow \\ &\qquad (implies(\beta_1,\alpha) = \beta_1 \vee \ldots \vee implies(\beta_{n_{k-1}},\alpha) = \beta_1 \vee \\ &\qquad\ implies(\beta_1,\alpha) = \beta_2 \vee \ldots \vee implies(\beta_{n_{k-1}},\alpha) = \beta_2 \vee \\ &\qquad\ \ldots \vee \\ &\qquad\ implies(\beta_1,\alpha) = \beta_{n_{k-1}} \vee \ldots \vee implies(\beta_{n_{k-1}},\alpha) = \beta_{n_{k-1}})]. \end{aligned}$$

Now, each of the disjuncts in the previous disjunction is of the form
$implies(\beta_i,\alpha) = \beta_j$.
Furthermore, if β_j is not of the form $implies(\gamma_1,\gamma_2)$, for some $\gamma_1,\gamma_2 \in \mathcal{L}(SL_0)$,
then by Lemma A.6, $SL^0 \vdash implies(\beta_i,\alpha) \neq \beta_j$.
Hence we are left with a disjunction where each disjunct is of the form,
$implies(\gamma_1,\alpha) = implies(\gamma_2,\gamma_3)$, for some $\gamma_1,\gamma_2,\gamma_3 \in \mathcal{L}(SL_0)$.
And, by EQ5, each of the disjuncts is equivalent to something of the form
$(\gamma_1 = \gamma_2 \wedge \alpha = \gamma_3)$, for some $\gamma_1,\gamma_2,\gamma_3 \in \mathcal{L}(SL_0)$.
The Equality Axiom and Lemma A.6 assure us that SL^0 can prove the truth or falsity of each of these disjuncts.
Hence we are left with,
$SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \longleftrightarrow (\alpha = \gamma_1 \vee \ldots \vee \alpha = \gamma_j)]$, for some $\gamma_1,\ldots,\gamma_j \in \mathcal{L}(SL_0)$.
Now, by FEED, $SL^0 \vdash (\forall\alpha)[Feed(\alpha,k) \longleftrightarrow (\alpha = \phi_1 \vee \alpha = \phi_2 \vee \ldots \vee \alpha = \phi_{l_k})]$.
Hence, by THM, $SL^0 \vdash (\forall\alpha)[K(k,\alpha) \longleftrightarrow (\alpha = \alpha_1 \vee \ldots \vee \alpha = \alpha_{n_k})]$,
where each α_i is either a ϕ_m or a γ_n. □

Lemma A.8 $$\begin{aligned} SL^0 \vdash (\forall\alpha)[Feed(\alpha,2) \longleftrightarrow (&\alpha = implies(P_0, implies(P_0, P_0)) \vee \\ &\alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \\ &\alpha = implies(P_1, implies(P_1, P_1)))]. \end{aligned}$$

There are exactly four wffs fed in at time-step 2.

Proof:

To show $$\begin{aligned} SL^0 \vdash (\forall\alpha)[(&\alpha = implies(P_0, implies(P_0, P_0)) \vee \\ &\alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \\ &\alpha = implies(P_1, implies(P_1, P_1))) \rightarrow Feed(\alpha,2)]. \end{aligned}$$

By LN3, $SL^0 \vdash l(implies(P_0, implies(P_0, P_0))) = l(P_0) + l(implies(P_0, P_0)) + 1$.
Again by LN3, $SL^0 \vdash l(implies(P_0, P_0)) = l(P_0) + l(P_0) + 1$.
Now, by PL1, $SL^0 \vdash Pl(P_0)$.
Hence, by LN1, $SL^0 \vdash l(P_0) = 0$.
Therefore, $SL^0 \vdash l(implies(P_0, implies(P_0, P_0))) = 2$.
By similar arguments, we get,

$SL^0 \vdash l(implies(P_0, implies(P_1, P_0))) = 2$.
$SL^0 \vdash l(implies(P_1, implies(P_0, P_1))) = 2$.
$SL^0 \vdash l(implies(P_1, implies(P_1, P_1))) = 2$.

Hence,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1))) \\ &\rightarrow l(\alpha) = 2.\end{aligned}$$

By P3, $SL^0 \vdash p(implies(P_0, implies(P_0, P_0))) = max(p(P_0), p(implies(P_0, P_0)))$.
Again by P3, $SL^0 \vdash p(implies(P_0, P_0)) = max(p(P_0), p(P_0))$.
Now, by P1, $SL^0 \vdash \alpha = P_0 \rightarrow p(\alpha) = 0$.
That is, $SL^0 \vdash p(P_0) = 0$.
Now, by MAX1, $SL^0 \vdash max(0, 0) = 0$.
Therefore, $SL^0 \vdash p(implies(P_0, implies(P_0, P_0))) = 0$.
By similar arguments, we get,

$SL^0 \vdash p(implies(P_0, implies(P_1, P_0))) = 1$.
$SL^0 \vdash p(implies(P_1, implies(P_0, P_1))) = 1$.
$SL^0 \vdash p(implies(P_1, implies(P_1, P_1))) = 1$.

Hence,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1))) \\ &\rightarrow p(\alpha) < 2.\end{aligned}$$

Now,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1))) \\ &\rightarrow (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta))).\end{aligned}$$

And, by AX, $SL^0 \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta))) \rightarrow Ax(\alpha)$.

Hence,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \\ &\alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \\ &\alpha = implies(P_1, implies(P_1, P_1))) \rightarrow Ax(\alpha).\end{aligned}$$

So we have,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1))) \\ &\rightarrow (Ax(\alpha) \wedge l(\alpha) = 2 \wedge p(\alpha) < 2).\end{aligned}$$

Hence,
$$\begin{aligned}SL^0 \vdash &(\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee \\ &\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1))) \\ &\rightarrow (Ax(\alpha) \wedge l(\alpha) \leq 2 \wedge p(\alpha) < 2).\end{aligned}$$

Then, by ALP, we have,
$SL^0 \vdash (\alpha = implies(P_0, implies(P_0, P_0)) \vee \alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee \alpha = implies(P_1, implies(P_1, P_1)))$
$\rightarrow Feed(\alpha, 2)$.

And, by *generalization*,
$SL^0 \vdash (\forall\alpha)[(\alpha = implies(P_0, implies(P_0, P_0)) \vee$
$\alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee$
$\alpha = implies(P_1, implies(P_1, P_1))) \rightarrow Feed(\alpha, 2)]$.

To show $SL^0 \vdash (\forall\alpha)[Feed(\alpha, 2) \rightarrow (\alpha = implies(P_0, implies(P_0, P_0)) \vee$
$\alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee$
$\alpha = implies(P_1, implies(P_1, P_1)))]$.

Let $\mathcal{T} = SL^0 + \{Feed(\alpha, 2)\}$.
Then, by ALP, we have, $\mathcal{T} \vdash (Ax(\alpha) \wedge l(\alpha) \leq 2 \wedge p(\alpha) < 2)$.
And, by Lemma A.4, $\mathcal{T} \vdash (Ax(\alpha) \wedge l(\alpha) = 2 \wedge p(\alpha) < 2)$.
Then, by Lemma A.9 (below),
$\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta))) \wedge l(\beta) = 0 \wedge l(\gamma) = 0)$.
Hence,
$\mathcal{T} \vdash \alpha = implies(b_0, implies(g_0, b_0))) \wedge l(b_0) = 0 \wedge l(g_0) = 0$, for some new constants b_0, g_0.
Now, by Lemma A.10 (below), $\mathcal{T} \vdash p(b_0) < 2 \wedge p(g_0) < 2$.
Then, by Lemma A.11 (below), $\mathcal{T} \vdash b_0 = P_0 \vee b_0 = P_1$ and $\mathcal{T} \vdash g_0 = P_0 \vee g_0 = P_1$.

Therefore,
$\mathcal{T} \vdash \alpha = implies(P_0, implies(P_0, P_0)) \vee$
$\alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee$
$\alpha = implies(P_1, implies(P_1, P_1))$.

Then, by the *Deduction theorem*,
$SL^0 \vdash Feed(\alpha, 2) \rightarrow (\alpha = implies(P_0, implies(P_0, P_0)) \vee$
$\alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee$
$\alpha = implies(P_1, implies(P_1, P_1)))$.

And, by *generalization*,
$SL^0 \vdash (\forall\alpha)[Feed(\alpha, 2) \rightarrow (\alpha = implies(P_0, implies(P_0, P_0)) \vee$
$\alpha = implies(P_0, implies(P_1, P_0)) \vee$
$\alpha = implies(P_1, implies(P_0, P_1)) \vee$
$\alpha = implies(P_1, implies(P_1, P_1)))]$.

□

Lemma A.9
$SL^0 \vdash (\forall\alpha)[(Ax(\alpha) \wedge l(\alpha) = 2) \rightarrow$
$(\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)) \wedge l(\beta) = 0 \wedge l(\gamma) = 0)]$.

Proof: Let $\mathcal{T} = SL^0 + \{Ax(\alpha), l(\alpha) = 2\}$.

Then, by AX,
$\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta))) \vee$
$\quad (\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)),$
$\qquad implies(implies(not(\gamma), \beta), \gamma))) \vee$
$\quad (\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma, \delta)),$
$\qquad implies(implies(\beta, \gamma), implies(\beta, \delta)))).$

Case 1:
Suppose $\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)),$
$\qquad implies(implies(not(\gamma), \beta), \gamma))).$

Then, by Lemma A.12 (below), $\mathcal{T} \vdash l(\alpha) \geq 7$.
But, by hypothesis, $\mathcal{T} \vdash l(\alpha) = 2$.

Therefore, it must be that,
$\mathcal{T} \nvdash (\exists\beta)(\exists\gamma)(\alpha = implies(implies(not(\gamma), not(\beta)),$
$\qquad implies(implies(not(\gamma), \beta), \gamma))).$

Case 2:
Suppose $\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma, \delta)),$
$\qquad implies(implies(\beta, \gamma), implies(\beta, \delta)))).$

Then, by Lemma A.13 (below), $\mathcal{T} \vdash l(\alpha) \geq 6$.
But, by hypothesis, $\mathcal{T} \vdash l(\alpha) = 2$.

Therefore, it must be that,
$\mathcal{T} \nvdash (\exists\beta)(\exists\gamma)(\exists\delta)(\alpha = implies(implies(\beta, implies(\gamma, \delta)),$
$\qquad implies(implies(\beta, \gamma), implies(\beta, \delta)))).$

Hence, $\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)))$.
Then, $\mathcal{T} \vdash \alpha = implies(b_0, implies(g_0, b_0))$, for some new constants b_0, g_0.
Now, by LN3, $\mathcal{T} \vdash l(implies(g_0, b_0)) = l(g_0) + l(b_0) + 1$.
Then, by LN3, $\mathcal{T} \vdash l(\alpha) = l(b_0) + [l(g_0) + l(b_0) + 1] + 1$.
Now, by hypothesis, $\mathcal{T} \vdash l(\alpha) = 2$.
Hence, $\mathcal{T} \vdash l(b_0) = 0$.
And, $\mathcal{T} \vdash l(g_0) = 0$.
So, $\mathcal{T} \vdash \alpha = implies(b_0, implies(g_0, b_0)) \wedge l(b_0) = 0 \wedge l(g_0) = 0$.
Hence, $\mathcal{T} \vdash (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)) \wedge l(\beta) = 0 \wedge l(\gamma) = 0)$.

Then, by the *Deduction theorem*,
$SL^0 \vdash (Ax(\alpha) \wedge l(\alpha) = 2) \rightarrow$
$\qquad (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)) \wedge l(\beta) = 0 \wedge l(\gamma) = 0)$.

And, by *generalization*,
$SL^0 \vdash (\forall\alpha)[(Ax(\alpha) \wedge l(\alpha) = 2) \rightarrow$
$\qquad (\exists\beta)(\exists\gamma)(\alpha = implies(\beta, implies(\gamma, \beta)) \wedge l(\beta) = 0 \wedge l(\gamma) = 0)]$.

□

Lemma A.10 $SL^0 \vdash (\forall\alpha)(\forall\beta)(\forall\gamma)[(p(\alpha) < 2 \wedge \alpha = implies(\beta, implies(\gamma, \beta))) \rightarrow$
$\qquad (p(\beta) < 2 \wedge p(\gamma) < 2)]$.

Proof: Let $\mathcal{T} = SL^0 + \{\alpha = implies(\beta, implies(\gamma, \beta)), p(\alpha) < 2\}$.
By P3, $\mathcal{T} \vdash p(\alpha) = max(p(\beta), p(implies(\gamma, \beta)))$.
Hence, $\mathcal{T} \vdash max(p(\beta), p(implies(\gamma, \beta))) < 2$.
From MAX1 and MAX2, $SL^0 \vdash (\forall i)(\forall j)(i \leq max(i, j) \wedge j \leq max(i, j))$.
Hence, $\mathcal{T} \vdash p(\beta) < 2$ and $\mathcal{T} \vdash p(implies(\gamma, \beta)) < 2$.
Then, by P3, $\mathcal{T} \vdash max(p(\gamma), p(\beta)) < 2$.
And we have, $\mathcal{T} \vdash p(\beta) < 2$ and $\mathcal{T} \vdash p(\gamma) < 2$.
Hence, $\mathcal{T} \vdash p(\beta) < 2 \wedge p(\gamma) < 2$.
Then, by the *Deduction theorem*,
$SL^0 \vdash (p(\alpha) < 2 \wedge \alpha = implies(\beta, implies(\gamma, \beta))) \rightarrow (p(\beta) < 2 \wedge p(\gamma) < 2)$.
Then, by *generalization*,
$SL^0 \vdash (\forall\alpha)(\forall\beta)(\forall\gamma)[(p(\alpha) < 2 \wedge \alpha = implies(\beta, implies(\gamma, \beta))) \rightarrow (p(\beta) < 2 \wedge p(\gamma) < 2)]$. □

Lemma A.11 $SL^0 \vdash (\forall\alpha)[(p(\alpha) < 2 \wedge l(\alpha) = 0) \rightarrow (\alpha = P_0 \vee \alpha = P_1)]$.

Proof: Let $\mathcal{T} = SL^0 + \{l(\alpha) = 0\}$.
By LN1, $\mathcal{T} \vdash Pl(\alpha)$.
By P1, $\mathcal{T} \vdash p(\alpha) = 0 \rightarrow \alpha = P_0$ and $\mathcal{T} \vdash p(\alpha) = 1 \rightarrow \alpha = P_1$.
Hence, $\mathcal{T} \vdash (p(\alpha) = 0 \vee p(\alpha) = 1) \rightarrow (\alpha = P_0 \vee \alpha = P_1)$.
And, by P4, $\mathcal{T} \vdash p(\alpha) < 2 \rightarrow (\alpha = P_0 \vee \alpha = P_1)$.
Then, by the *Deduction theorem*, $SL^0 \vdash l(\alpha) = 0 \rightarrow [p(\alpha) < 2 \rightarrow (\alpha = P_0 \vee \alpha = P_1)]$.
Hence, $SL^0 \vdash (p(\alpha) < 2 \wedge l(\alpha) = 0) \rightarrow (\alpha = P_0 \vee \alpha = P_1)$.
And, by *generalization*, $SL^0 \vdash (\forall\alpha)[(p(\alpha) < 2 \wedge l(\alpha) = 0) \rightarrow (\alpha = P_0 \vee \alpha = P_1)]$. □

Lemma A.12 $SL^0 \vdash (\forall\alpha)(\forall\beta)[l(implies(implies(not(\beta), not(\alpha)),$
$implies(implies(not(\beta), \alpha), \beta))) \geq 7]$.

Proof:

By LN3,
$SL^0 \vdash l(implies(implies(not(\beta), not(\alpha)), implies(implies(not(\beta), \alpha), \beta))) =$
$l(implies(not(\beta), not(\alpha))) + l(implies(implies(not(\beta), \alpha), \beta)) + 1$.

and,
$SL^0 \vdash l(implies(not(\beta), not(\alpha))) = l(not(\beta)) + l(not(\alpha)) + 1$
and,
$SL^0 \vdash l(implies(implies(not(\beta), \alpha), \beta)) = l(implies(not(\beta), \alpha)) + l(\beta) + 1$
and,
$SL^0 \vdash l(implies(not(\beta), \alpha)) = l(not(\beta)) + l(\alpha) + 1$.

Then, by LN2,
$SL^0 \vdash l(implies(implies(not(\beta), not(\alpha)), implies(implies(not(\beta), \alpha), \beta))) =$
$[[l(\beta) + 1] + [l(\alpha) + 1] + 1] + [[[l(\beta) + 1] + l(\alpha) + 1] + l(\beta) + 1] + 1$.

And, by LN4,
$SL^0 \vdash l(implies(implies(not(\beta), not(\alpha)), implies(implies(not(\beta), \alpha), \beta))) \geq 7$.

Then, by *generalization*,
$SL^0 \vdash (\forall\alpha)(\forall\beta)[l(implies(implies(not(\beta), not(\alpha)),$
$implies(implies(not(\beta), \alpha), \beta))) \geq 7]$.

□

Lemma A.13
$SL^0 \vdash (\forall\alpha)(\forall\beta)(\forall\gamma)[l(implies(implies(\alpha, implies(\beta,\gamma)),$
$implies(implies(\alpha,\beta), implies(\alpha,\gamma)))) \geq 6].$

Proof:

By LN3,
$SL^0 \vdash l(implies(implies(\alpha, implies(\beta,\gamma)),$
$implies(implies(\alpha,\beta), implies(\alpha,\gamma)))) =$
$l(implies(\alpha, implies(\beta,\gamma))) + l(implies(implies(\alpha,\beta), implies(\alpha,\gamma))) + 1$

and,
$SL^0 \vdash l(implies(\alpha, implies(\beta,\gamma))) = l(\alpha) + l(implies(\beta,\gamma)) + 1$

and,
$SL^0 \vdash l(implies(implies(\alpha,\beta), implies(\alpha,\gamma))) =$
$l(implies(\alpha,\beta)) + l(implies(\alpha,\gamma)) + 1.$

Then, by LN3,
$SL^0 \vdash l(implies(implies(\alpha, implies(\beta,\gamma)),$
$implies(implies(\alpha,\beta), implies(\alpha,\gamma)))) =$
$[l(\alpha) + [l(\beta) + l(\gamma) + 1] + 1]+$
$[[l(\alpha) + l(\beta) + 1] + [l(\alpha) + l(\gamma) + 1] + 1] + 1.$

And, by LN4,
$SL^0 \vdash l(implies(implies(\alpha, implies(\beta,\gamma)),$
$implies(implies(\alpha,\beta), implies(\alpha,\gamma)))) \geq 6.$

Then, by *generalization*,
$SL^0 \vdash (\forall\alpha)(\forall\beta)(\forall\gamma)[l(implies(implies(\alpha, implies(\beta,\gamma)),$
$implies(implies(\alpha,\beta), implies(\alpha,\gamma)))) \geq 6].$

□

A.3.2 The Main Theorem

We can now prove the following result.

Theorem A.14 (Analytic Completeness Theorem) *For each $i \in \mathcal{N}$, and for each $\alpha \in \mathcal{L}(SL_0)$,*

$$\textit{either } SL^0 \vdash K(i,\alpha) \qquad \textit{or} \qquad SL^0 \vdash \neg K(i,\alpha).$$

SL^0 can characterize exactly what has and has not been proved at any given time i.

Proof: By weak induction on i.
Base case: To show, for any $\alpha \in \mathcal{L}(SL_0)$, $SL^0 \vdash K(0,\alpha)$ or
$SL^0 \vdash \neg K(0,\alpha)$.

By the Newborn Lemma, $SL^0 \vdash (\forall\alpha)(\neg K(0,\alpha))$.
Therefore, for any α, we have $SL^0 \vdash \neg K(0,\alpha)$.

Assume for all $\alpha \in \mathcal{L}(SL_0)$, either $SL^0 \vdash K(k-1,\alpha)$ or $SL^0 \vdash \neg K(k-1,\alpha)$, where $k > 0$.
To show for any $\alpha \in \mathcal{L}(SL_0)$, either $SL^0 \vdash K(k,\alpha)$ or $SL^0 \vdash \neg K(k,\alpha)$.
Fix α and suppose $SL^0 \not\vdash K(k,\alpha)$. *To show* $SL^0 \vdash \neg K(k,\alpha)$.

Suppose $SL^0 \vdash K(k-1,\alpha)$.
Then, by the Monotonicity Lemma, $SL^0 \vdash K(k,\alpha)$. $\longrightarrow\!\longleftarrow$
Therefore, $SL^0 \nvdash K(k-1,\alpha)$.
Now, by hypothesis, $SL^0 \nvdash K(k,\alpha)$.
Hence, by THM, $SL^0 \nvdash Feed(\alpha,k) \vee Mp(\alpha,k)$.
That is, $SL^0 \nvdash Feed(\alpha,k)$, and $SL^0 \nvdash Mp(\alpha,k)$.
To show, $SL^0 \vdash \neg Feed(\alpha,k)$.

Suppose k=1.

By Lemma 3 and THM, $SL^0 \vdash \neg[Feed(\alpha,1) \vee Mp(\alpha,1)]$.
Hence, $SL^0 \vdash \neg Feed(\alpha,1)$.

Suppose $k > 1$.

By FEED, $SL^0 \vdash Feed(\alpha,k) \longleftrightarrow (\alpha = \phi_1 \vee \alpha = \phi_2 \vee \ldots \vee \alpha = \phi_{l_k})$.
By the Equality and Inequality Lemmas, SL^0 can prove the truth or falsity of each of these disjuncts; hence, SL^0 can prove the truth or falsity of $Feed(\alpha,k)$.
But, $SL^0 \nvdash Feed(\alpha,k)$.
Therefore, it must be that, $SL^0 \vdash \neg Feed(\alpha,k)$

To show $SL^0 \vdash \neg Mp(\alpha,k)$.

Since $SL^0 \nvdash Mp(\alpha,k)$, by MP,

$$SL^0 \nvdash (\exists j)(\exists l)(\exists\beta)(K(j,\beta) \wedge K(l, implies(\beta,\alpha)) \wedge j < k \wedge l < k). \qquad \text{(A.1)}$$

And, in particular, $SL^0 \nvdash (\exists\beta)(K(k-1,\beta) \wedge K(k-1, implies(\beta,\alpha)))$.
Case 1: k=1

By the Newborn Lemma, we have,

$$SL^0 \vdash \neg(\exists\beta)(K(k-1,\beta) \wedge K(k-1, implies(\beta,\alpha))). \qquad \text{(A.2)}$$

Case 2: k=2

By Lemma A.3, we again have equation (A.2).

Case 3: $k > 2$

By the Boundedness Lemma,
$SL^0 \nvdash (\exists\beta)[\,(\beta = \beta_1 \vee \ldots \vee \beta = \beta_{n_{k-1}})\wedge$
$(implies(\beta,\alpha) = \beta_1 \vee \ldots \vee implies(\beta,\alpha) = \beta_{n_{k-1}})]$,
where the $\beta_1 \ldots \beta_{n_{k-1}}$ are those that are asserted to exist in the lemma.
But this is true iff,
$SL^0 \nvdash implies(\beta_1,\alpha) = \beta_1 \vee \ldots \vee$
$implies(\beta_{n_{k-1}},\alpha) = \beta_1 \vee$
$implies(\beta_1,\alpha) = \beta_2 \vee \ldots \vee$
$implies(\beta_{n_{k-1}},\alpha) = \beta_2 \vee \ldots \vee$
$implies(\beta_1,\alpha) = \beta_{n_{k-1}} \vee \ldots \vee$
$implies(\beta_{n_{k-1}},\alpha) = \beta_{n_{k-1}}$.
That is,
$SL^0 \nvdash implies(\beta_1,\alpha) = \beta_1$, and
$SL^0 \nvdash implies(\beta_2,\alpha) = \beta_1$, and ... and
$SL^0 \nvdash implies(\beta_{n_{k-1}},\alpha) = \beta_{n_{k-1}}$.
By the Equality Axiom,
$SL^0 \vdash \gamma = \delta$, if γ and δ are the same;
hence it must be true that,
$implies(\beta_1,\alpha) \neq \beta_1$, and

$implies(\beta_2, \alpha) \neq \beta_1$, and ... and
$implies(\beta_{n_{k-1}}, \alpha) \neq \beta_{n_{k-1}}$.
By the Inequality Lemma, then,
$SL^0 \vdash implies(\beta_1, \alpha) \neq \beta_1$, and
$SL^0 \vdash implies(\beta_2, \alpha) \neq \beta_1$, and ... and
$SL^0 \vdash implies(\beta_{n_{k-1}}, \alpha) \neq \beta_{n_{k-1}}$.
Hence,
$SL^0 \vdash \neg(implies(\beta_1, \alpha) = \beta_1 \vee \ldots \vee implies(\beta_{n_{k-1}}, \alpha) = \beta_{n_{k-1}})$.
Therefore,
$SL^0 \vdash \neg(\exists\beta)[\ (\beta = \beta_1 \vee \ldots \vee \beta = \beta_{n_{k-1}})\wedge$
$(implies(\beta, \alpha) = \beta_1 \vee \ldots \vee implies(\beta, \alpha) = \beta_{n_{k-1}})]$,
Therefore, since $\beta_1, \ldots, \beta_{n_{k-1}}$ are those that are asserted to exist in the Boundedness Lemma, the lemma gives us equation (A.2).

So, for any value of $k > 0$, we arrive at equation (A.2).
Now, since $SL^0 \not\vdash K(k-1, \alpha)$, by the inductive hypothesis,
$SL^0 \vdash \neg K(k-1, \alpha)$.
Hence, by THM, $SL^0 \vdash \neg[Feed(\alpha, k-1) \vee Mp(\alpha, k-1)]$.
Thus, $SL^0 \vdash \neg Mp(\alpha, k-1)$.
Then, by MP and equation (A.2), we have,
$SL^0 \vdash \neg(\exists j)(\exists l)(\exists\beta)(K(j, \beta) \wedge K(l, implies(\beta, \alpha)) \wedge j < k \wedge l < k)$.
And, by MP, $SL^0 \vdash \neg Mp(\alpha, k)$.

Since both $SL^0 \vdash \neg Mp(\alpha, k)$ and $SL^0 \vdash \neg Feed(\alpha, k)$,
we have $SL^0 \vdash \neg[Feed(\alpha, k) \vee Mp(\alpha, k)]$.
By THM, this gives us, $SL^0 \vdash \neg K(k, \alpha)$.
So, if $SL^0 \not\vdash K(k, \alpha)$, then $SL^0 \vdash \neg K(k, \alpha)$.

Thus, for any $\alpha \in \mathcal{L}(SL_0)$, either $SL^0 \vdash K(k, \alpha)$ or $SL^0 \vdash \neg K(k, \alpha)$.
□

A.4 Other results

In this section we prove the theorems appearing in Section 4.2.2 that were not required for the proof of *Analytic Completeness*.

Theorem A.15 $SL^0 \vdash (\forall\alpha)(\forall\beta)[[Taut(\alpha) \wedge Taut(implies(\alpha, \beta))] \rightarrow Taut(\beta)]$.

Proof: Let $\mathcal{T} = SL^0 + \{Taut(\alpha), Taut(implies(\alpha, \beta))\}$.
Then by TAUT1, $\mathcal{T} \vdash (\forall x)[Tfcn(x) \rightarrow True(x, implies(\alpha, \beta))]$.
And by TFCN1, $\mathcal{T} \vdash (\forall x)[Tfcn(x) \rightarrow (True(x, \beta) \vee \neg True(x, \alpha))]$.
But, by TAUT1, $\mathcal{T} \vdash (\forall x)[Tfcn(x) \rightarrow True(x, \alpha)]$.
Hence, $\mathcal{T} \vdash (\forall x)[Tfcn(x) \rightarrow True(x, \beta)]$.
Then, by TAUT1, $\mathcal{T} \vdash Taut(\beta)$.
Hence, by the *Deduction theorem*, we have,
$SL^0 \vdash [Taut(\alpha) \wedge Taut(implies(\alpha, \beta))] \rightarrow Taut(\beta)$.
And, by *generalization*,
$SL^0 \vdash (\forall\alpha)(\forall\beta)[[Taut(\alpha) \wedge Taut(implies(\alpha, \beta))] \rightarrow Taut(\beta)]$. □

Theorem A.16 *For any $i \in \mathcal{N}$, $SL^0 \vdash (\forall\alpha)[K(i, \alpha) \rightarrow Taut(\alpha)]$.*
The only wffs an agent can prove are those that are tautologies.

Proof: By strong induction on i.

Base case: To show, $SL^0 \vdash (\forall\alpha)[K(0,\alpha) \rightarrow Taut(\alpha)]$.

By Lemma A.1, $SL^0 \vdash (\forall\alpha)(\neg K(0,\alpha))$.
Hence, trivially, $SL^0 \vdash (\forall\alpha)[K(0,\alpha) \rightarrow Taut(\alpha)]$.

Assume for $i < k$, $k > 0$, $SL^0 \vdash (\forall\alpha)[K(i,\alpha) \rightarrow Taut(\alpha)]$.
To show $SL^0 \vdash (\forall\alpha)[K(k,\alpha) \rightarrow Taut(\alpha)]$.
By ALP, $SL^0 \vdash (\forall\alpha)[Feed(\alpha,k) \rightarrow Ax(\alpha)]$.
And by TAUT2,

$$SL^0 \vdash (\forall\alpha)[Feed(\alpha,k) \rightarrow Taut(\alpha)]. \tag{A.3}$$

By MP, $SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \rightarrow (\exists l)(\exists m)(\exists\beta)(K(l,\beta) \wedge K(m, implies(\beta,\alpha)) \wedge l < k \wedge m < k)]$.
[Note that, for $j = l$ or m, $(j < k \rightarrow j = 0 \vee j = 1 \vee \ldots \vee j = (k-1))$.]
Then, by the inductive hypothesis,
$SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \rightarrow (\exists\beta)[Taut(\beta) \wedge Taut(implies(\beta,\alpha))]]$.
Then, by Theorem A.15,

$$SL^0 \vdash (\forall\alpha)[Mp(\alpha,k) \rightarrow Taut(\alpha)]. \tag{A.4}$$

By THM, and equations (A.3) and (A.4), we get,
$SL^0 \vdash (\forall\alpha)[K(k,\alpha) \rightarrow Taut(\alpha)]$. □

Lemma A.17 *$SL^0 \vdash \neg Taut(P_i)$, for any propositional letter $P_i \in \mathcal{L}(SL_0)$.*
P_i is not a tautology, for all propositional letters P_i.

Proof: By PL1, $SL^0 \vdash Pl(P_i)$.
Then by PL2, $SL^0 \vdash (\exists x)(Tfcn(x) \wedge \neg True(x, P_i))$.
And by TAUT1, $SL^0 \vdash \neg Taut(P_i)$. □

Corollary A.18 *$SL^0 \vdash \neg K(i, P_j)$, for any $i, j \in \mathcal{N}$.*
The agent can never prove P_j, where P_j is a propositional letter.

Proof: Follows immediately from Theorem A.16 and Lemma A.17. □

B Proofs of Theorems about $\mathbf{SL_7}$

B.1 Theorem 5.4

Theorem B.1 *$SL_7(OBS, INF_B)$ is step-wise consistent if OBS is both valid and Now-free.*

Proof: We show $SL_7(OBS, INF_B)$ has a step-model, and apply Theorem 3.16. Let M_i be such that:

1. (MODEL-NOW) $M_i \models Now(x)$ iff $x = i$.
2. (MODEL-K) $M_i \models K(j, \alpha)$ iff $\vdash_j \alpha$.
3. (MODEL-P) $M_i \not\models P(x_1, \ldots, x_n)$ if P is a predicate other than Now or K.

We show $M =< M_0, M_1, \ldots, M_i, \ldots >$ a step-model for $SL_7(OBS, INF_B)$ by induction on the index. For each index i, we want to show the following:[1]

1. (HYP.CONTRA) $Contra(\alpha, \beta, \gamma) \notin \vdash_i$.
2. (HYP.NOW) If $M_i \models K(i, \alpha)$ and α is not Now-free, then $\alpha = Now(i)$.
3. (HYP.MODEL) $M_i \models \alpha$ if $\vdash_i \alpha$.
4. (HYP.CONSISTENT) $\vdash_i$ is consistent.

Base case: $i = 0$

1. (HYP.CONTRA) This is true since $\vdash_0$ is empty.
2. (HYP.NOW) By (MODEL-K), $M_0 \models K(0, \alpha)$ iff $\vdash_0 \alpha$.
   Since $\vdash_0$ is empty, $M_0 \not\models K(0, \alpha)$ for any α.
   Therefore, this hypothesis is trivially true.
3. (HYP.MODEL) Since $\vdash_0$ is empty, this hypothesis is trivially true.
4. (HYP.CONSISTENT) $\vdash_0$ is consistent since it is empty.

Assume Hypotheses (HYP.CONTRA), (HYP.NOW), (HYP.MODEL), and (HYP.CONSISTENT) for $i - 1$. We must show these are true for i.

1. (HYP.CONTRA) To show $Contra(\alpha, \beta, \gamma) \notin \vdash_i$.
   By INF_B, $Contra(\alpha, \beta, \gamma) \in \vdash_i$ only thru the Rules 1-7. But:
   (a) Rule 1 will not bring in any wffs of the form $Contra(\alpha, \beta, \gamma)$.
   (b) Rule 2 will not bring in any wffs of the form $Contra(\alpha, \beta, \gamma)$.
   (c) Suppose $\delta, \delta \rightarrow Contra(\alpha, \beta, \gamma) \in \vdash_{i-1}$.
   Then, by Hyp. (HYP.MODEL), $M_{i-1} \models \delta$ and
   $M_{i-1} \models \delta \rightarrow Contra(\alpha, \beta, \gamma)$.
   Hence, since M_{i-1} is an interpretation, $M_{i-1} \models Contra(\alpha, \beta, \gamma)$.
   But, by (MODEL-P), $M_{i-1} \not\models Contra(\alpha, \beta, \gamma)$. $\longrightarrow\!\longleftarrow$
   Thus both δ and $\delta \rightarrow Contra(\alpha, \beta, \gamma)$ cannot be $\in \vdash_{i-1}$.
   Therefore Rule 3 will not produce $Contra(\alpha, \beta, \gamma)$ at step i.

[1]A step-model requires that $M_i \models K(i, \alpha)$ iff $\vdash_i \alpha$. This we know to be true $\forall i$ directly from (MODEL-K).

(d) Suppose $P_1a, \ldots, P_na, \forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Contra(\alpha, \beta, \gamma)] \in \vdash_{i-1}$.
Then, by Hyp. (HYP.MODEL), $M_{i-1} \models P_1a$ and, …, and $M_{i-1} \models P_na$ and $M_{i-1} \models \forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Contra(\alpha, \beta, \gamma)]$.
Hence, since M_{i-1} is an interpretation, $M_{i-1} \models Contra(\alpha, \beta, \gamma)$.
But, by (MODEL-P), $M_{i-1} \not\models Contra(\alpha, \beta, \gamma)$. $\longrightarrow\!\longleftarrow$
Thus $P_1a, \ldots, P_na, \forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Contra(\alpha, \beta, \gamma)]$ cannot all be $\in \vdash_{i-1}$.
Therefore Rule 4 will not produce $Contra(\alpha, \beta, \gamma)$ at step i.

(e) Rule 5 will not bring in any wffs of the form $Contra(\alpha, \beta, \gamma)$.

(f) By inductive hyp. (HYP.CONSISTENT), $\vdash_{i-1}$ is consistent.
Thus α and $\neg\alpha$ cannot both be $\in \vdash_{i-1}$.
Therefore, Rule 6 will not apply.

(g) By the inductive hypothesis, $Contra(\alpha, \beta, \gamma) \notin \vdash_{i-1}$.
Therefore Rule 7 will not bring in any wffs of the form $Contra(\alpha, \beta, \gamma)$.

Therefore $Contra(\alpha, \beta, \gamma) \notin \vdash_i$.

2. (HYP.NOW) Suppose $M_i \models K(i, \alpha)$ and α is not Now-free. To show $\alpha = Now(i)$.
From (MODEL-K), $\vdash_i \alpha$.
By INF_B, either:

(a) $\alpha = Now(i)$.

(b) $\alpha \in OBS(i-1)$. Since OBS is Now-free, Now doesn't appear in α. $\longrightarrow\!\longleftarrow$

(c) $\beta, \beta \rightarrow \alpha \in \vdash_{i-1}$.
Then, by (MODEL-K), $M_{i-1} \models K(i-1, \beta \rightarrow \alpha)$.
Since α is not Now-free, $\beta \rightarrow \alpha$ is not Now-free.
But by the inductive hypothesis, if $\beta \rightarrow \alpha$ is not Now-free, then $\beta \rightarrow \alpha$ is $Now(i-1)$. $\longrightarrow\!\longleftarrow$

(d) $\alpha = Qa$ and $P_1a, \ldots, P_na, \forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Q(x)] \in \vdash_{i-1}$.
Since α contains Now, Q must be Now.
Then by (MODEL-K), $M_{i-1} \models K(i-1, \forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Now(x)])$.
But, by the inductive hypothesis, $\forall x[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Now(x)])$ must be $Now(i-1)$. $\longrightarrow\!\longleftarrow$

(e) $\alpha = \neg K(i-1, \beta)$ and $\beta \notin \vdash_{i-1}$ and $\gamma \in \vdash_{i-1}$, where β is a closed sub-formula of γ.
Since Now appears in α, Now must also appear in β, and thus must also appear in γ.
Now, by (MODEL-K), $M_{i-1} \models K(i-1, \gamma)$.
But then, by the inductive hypothesis, $\gamma = Now(i-1)$. Hence $\beta = \gamma$.
But then, $\beta \in \vdash_{i-1}$. $\longrightarrow\!\longleftarrow$

(f) $\alpha = Contra(i-1, \beta, \neg\beta)$.
But by (HYP.CONTRA), $Contra(\alpha, \beta, \gamma) \notin \vdash_i$. $\longrightarrow\!\longleftarrow$

(g) $\alpha \in \vdash_{i-1}$ and $\alpha \neq Now(\beta)$ and $Contra(i-2, \alpha, \gamma) \notin \vdash_{i-1}$ and $Contra(i-2, \gamma, \alpha) \notin \vdash_{i-1}$.
Then, by (MODEL-K), $M_{i-1} \models K(i-1, \alpha)$.
Then by the inductive hypothesis, $\alpha = Now(i-1)$. $\longrightarrow\!\longleftarrow$

Therefore, if $M_i \models K(i, \alpha)$ and α is not Now-free, then $\alpha = Now(i)$.

3. (HYP.MODEL) Let $\alpha \in \vdash_i$. To show $M_i \models \alpha$.
By INF_B, either:

(a) $\alpha = Now(i)$.
Then by (MODEL-NOW), $M_i \models \alpha$.

(b) $\alpha \in OBS(i-1)$. Then α is valid.
Hence α is true in any interpretation; and in particular, $M_i \models \alpha$.

(c) $\beta, \beta \rightarrow \alpha \in \vdash_{i-1}$.
Then, by the inductive hypothesis, $M_{i-1} \models \beta$ and $M_{i-1} \models \beta \rightarrow \alpha$.
Hence, since M_{i-1} is an interpretation, $M_{i-1} \models \alpha$.
Now, by (MODEL-K), $M_{i-1} \models K(i-1, \beta \rightarrow \alpha)$.
And by Hyp. (HYP.NOW), if Now appears in $\beta \rightarrow \alpha$, then $\beta \rightarrow \alpha$ is $Now(i-1)$. $\longrightarrow\!\!\longleftarrow$
Therefore, $\beta \rightarrow \alpha$ is Now-free; hence α is Now-free.
Then by Lemma B.2, $M_i \models \alpha$.

(d) $\alpha = Qa$ and $P_1 a, \ldots, P_n a, \forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)] \in \vdash_{i-1}$.
Then, by the inductive hypothesis,
$M_{i-1} \models P_1 a$ and ... and $M_{i-1} \models P_n a$ and $M_{i-1} \models \forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)]$.
Hence, since M_{i-1} is an interpretation, $M_{i-1} \models Qa$, i.e. $M_{i-1} \models \alpha$.
Now, by (MODEL-K), $M_{i-1} \models K(i-1, \forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)])$.
And by Hyp. (HYP.NOW), if Now appears in $\forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)]$,
then $\forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)]$ is $Now(i-1)$. $\longrightarrow\!\!\longleftarrow$
Therefore, $\forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Q(x)]$ is Now-free; and in particular, Q is not Now.
Hence α is Now-free.
Then by Lemma B.2, $M_i \models \alpha$.

(e) $\alpha = \neg K(i-1, \beta)$ and $\beta \notin \vdash_{i-1}$ and $\gamma \in \vdash_{i-1}$, where β is a closed sub-formula of γ.
By (MODEL-K), $M_{i-1} \not\models K(i-1, \beta)$.
And since M_{i-1} is an interpretation, $M_{i-1} \models \neg K(i-1, \beta)$, i.e. $M_{i-1} \models \alpha$.
To show α is Now-free, it is sufficient to show β is Now-free.
Now, by (MODEL-K), $M_{i-1} \models K(i-1, \gamma)$.
And by Hyp. (HYP.NOW), if Now appears in γ, then $\gamma = Now(i-1)$.
Now β is a closed sub-formula of γ, hence $\beta = Now(i-1)$. Then $\beta \in \vdash_{i-1}$. $\longrightarrow\!\!\longleftarrow$
Therefore γ is Now-free; hence, β is Now-free.
Since β is Now-free, α is also Now-free.
Then by Lemma B.2, $M_i \models \alpha$.

(f) $\alpha = Contra(i-1, \beta, \neg\beta)$ and $\beta, \neg\beta \in \vdash_{i-1}$.
But by Hyp. (HYP.CONSISTENT), we cannot have both $\beta \in \vdash_{i-1}$ and $\neg\beta \in \vdash_{i-1}$.
Therefore $\alpha \neq Contra(i-1, \beta, \neg\beta)$. $\longrightarrow\!\!\longleftarrow$

(g) $\alpha \in \vdash_{i-1}$ and $\alpha \neq Now(\beta)$ and $Contra(i-2, \alpha, \gamma) \notin \vdash_{i-1}$ and $Contra(i-2, \gamma, \alpha) \notin \vdash_{i-1}$.
Now, by (MODEL-K), $M_{i-1} \models K(i-1, \alpha)$.
And by Hyp. (HYP.NOW), if Now appears in α, $\alpha = Now(i-1)$. $\longrightarrow\!\!\longleftarrow$
Therefore, α is Now-free.
Now by the inductive hypothesis, $M_{i-1} \models \alpha$.
Then by Lemma B.2, $M_i \models \alpha$.

Therefore, if $\alpha \in \vdash_i$, then $M_i \models \alpha$.

4. (HYP.CONSISTENT) To show $\vdash_i$ is consistent.
Suppose $\vdash_i$ is inconsistent. Then there exist wffs $\alpha_1, \ldots, \alpha_n \in \vdash_i$ which are mutually inconsistent.
By Hyp. (HYP.MODEL), $M_i \models \alpha_1$ and ... and $M_i \models \alpha_n$.
But since M_i is an interpretation, $\alpha_1, \ldots, \alpha_n$ cannot be mutually inconsistent.
Therefore, $\vdash_i$ is consistent.

Therefore, by induction we have shown that (HYP.CONTRA), (HYP.NOW), (HYP.MODEL), and (HYP.CONSISTENT) hold for all i.
Now, (HYP.MODEL) shows that $M =< M_0, M_1, \ldots, M_i, \ldots >$ is a step-model for $SL_7(OBS, INF_B)$.
And by Theorem 3.16, $SL_7(OBS, INF_B)$ is step-wise consistent.
(We also have step-wise consistency directly from (HYP.CONSISTENT).) □

Lemma B.2 *$M_i \models \alpha$ if $M_{i-1} \models \alpha$ and α is Now-free.*

Proof: By (MODEL:K), if a wff $K(j, \beta)$ is true in some M_i, then it is true in every M_i.
Likewise, if a wff $K(j, \beta)$ is false in some M_i, then it is false in every M_i.
By (MODEL:P), wffs $P(x_1, \ldots, x_n)$, where the predicate letter P is neither Now nor K, is false in every M_i.
It follows that every Now-free wff α will either be true in every M_i or false in every M_i. For such wffs will be built out of wffs $K(j, \beta)$ and $P(t_1, \ldots, t_n)$ whose truth-values do not change with i.
Therefore, if $M_{i-1} \models \alpha$ and α is Now-free, then $M_i \models \alpha$. □

C Sample Runs of Programs

This appendix contains the example queries run on the various implementations of the step-logics.

C.1 SL^0

The following is an actual scenario of queries. The query "$state(i, \alpha)$" asks for a wff which is true at step i, i.e., $state(i, \alpha)$ corresponds to $K(i, \alpha)$.

```
?- state(0,X).                              /* Find an X proven at step 0.*/
no                                          /* There are no wffs proven
                                            at step 0.*/


?- state(5,X).                              /* Find an X proven at step 5.*/
X = imp(p(0),imp(p(0),p(0))) →;             /* X = P0 → (P0 → P0).
                                            ';' requests another soln.*/
no                                          /* There are no more wffs proven
                                            at step 5.*/


?- state(6,X).
X = imp(p(0),imp(p(1),p(0))) →;             /* X = P0 → (P1 → P0).*/
X = imp(p(0),imp(neg(p(0)),p(0))) →;        /* X = P0 → (¬P0 → P0).*/
X = imp(p(0),imp(p(0),p(0))) →;             /* X = P0 → (P0 → P0).*/
no


?- state(7,X).
X = imp(p(0),imp(p(2),p(0))) →;
X = imp(p(0),imp(imp(p(0),p(0)),p(0))) →;
X = imp(p(0),imp(neg(p(1)),p(0))) →;
X = imp(p(0),imp(neg(neg(p(0))),p(0))) →;
X = imp(p(1),imp(p(0),p(1))) →;
X = imp(neg(p(0)),imp(p(0),neg(p(0)))) →;
X = imp(p(0),imp(p(1),p(0))) →;
X = imp(p(0),imp(neg(p(0)),p(0))) →;
X = imp(p(0),imp(p(0),p(0))) →;
no


?- state(N,imp(p(0),imp(p(0),p(0)))).       /* At what step is P0 → (P0 → P0) proved?*/
N = 5 →;                                    /* Step 5 is the 1st step in which P0 → (P0 → P0)
                                            is proved.*/
N = 6 →                                     /* CR means no more solutions are wanted.*/
yes                                         /* The query was satisfied.*/


?- state(13,imp(imp(X,imp(X,X)),imp(X,imp(X,imp(X,X))))).
X = p(0)
yes
```

```
?- state(10,p(0)).                          /* Is p(0) proved at step 10?*/
no


?- state(12,X).
X = imp(p(0),imp(p(7),p(0))) →;
X = imp(p(0),imp(imp(p(0),p(5)),p(0))) →;
X = imp(p(0),imp(imp(p(0),imp(p(0),p(3))),p(0))) →
yes
```

C.2 Two-wise-men

The following is an actual scenario of queries. The query ''*step*(i)'' asks for all wffs which are true at step i. The predicates B_1, W_2, $s(i)$, $K_1(i,x)$, and $K_2(i,x)$ are represented as $b1$, $w2$, $s(i)$, $k1(i,x)$, and $k2(i,x)$, respectively. Valid formulas are built up from the predicates using *neg* and *imp* in the expected way. Any wff with a free variable (something of the form ''_dd'', where ''dd'' is any numeral) has an implicit universal quantifier. This trick of leaving off the universal quantifier made the implementation a little easier, and seemed to cause no problems.

Note that at step 10 the agent has indeed proven that his spot is white (i.e. W_1).

```
| ?- step(0).
yes

| ?- step(1).
imp(k2(_19,imp(_20,_21)),imp(k2(_19,_20),k2(s(_19),_21)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_19),w2)),neg(k2(_19,w2)))
imp(neg(k1(s(_19),u(_19,w2))),neg(u(_19,w2)))
yes

| ?- step(2).
imp(k2(_19,imp(_20,_21)),imp(k2(_19,_20),k2(s(_19),_21)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_19),w2)),neg(k2(_19,w2)))
imp(neg(k1(s(_19),u(_19,w2))),neg(u(_19,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
yes

| ?- step(3).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
yes
```

```
| ?- step(4).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
yes

| ?- step(5).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
yes

| ?- step(6).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k2(s(0),w2))
neg(u(s(s(s(0))),w2))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
yes

| ?- step(7).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k2(s(0),w2))
neg(u(s(s(s(0))),w2))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
neg(k2(s(s(0)),w2))
neg(u(s(s(s(s(0)))),w2))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w2)))
yes

| ?- step(8).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k2(s(0),w2))
neg(u(s(s(s(0))),w2))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
neg(k2(s(s(0)),w2))
neg(u(s(s(s(s(0)))),w2))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w2)))
neg(k2(s(s(s(0))),w2))
neg(u(s(s(s(s(s(0))))),w2))
neg(k2(s(0),b1))
neg(k1(s(s(s(s(s(s(s(0))))))),u(s(s(s(s(s(s(0)))))),w2)))
yes

| ?- step(9).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k2(s(0),w2))
neg(u(s(s(s(0))),w2))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
neg(k2(s(s(0)),w2))
neg(u(s(s(s(s(0)))),w2))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w2)))
neg(k2(s(s(s(0))),w2))
neg(u(s(s(s(s(s(0))))),w2))
neg(k2(s(0),b1))
neg(k1(s(s(s(s(s(s(s(0))))))),u(s(s(s(s(s(s(0)))))),w2)))
neg(k2(s(s(s(s(0)))),w2))
neg(u(s(s(s(s(s(s(0)))))),w2))
neg(b1)
neg(k1(s(s(s(s(s(s(s(s(0)))))))),u(s(s(s(s(s(s(s(0))))))),w2)))
yes

| ?- step(10).
imp(k2(_66,imp(_67,_68)),imp(k2(_66,_67),k2(s(_66),_68)))
k2(s(0),imp(b1,w2))
imp(b1,k2(s(0),b1))
imp(neg(b1),w1)
imp(neg(u(s(_66),w2)),neg(k2(_66,w2)))
imp(neg(k1(s(_66),u(_66,w2))),neg(u(_66,w2)))
imp(k2(s(0),b1),k2(s(s(0)),w2))
neg(k1(s(s(0)),u(s(0),w2)))
neg(u(s(0),w2))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k2(0,w2))
neg(u(s(s(0)),w2))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k2(s(0),w2))
neg(u(s(s(s(0))),w2))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
neg(k2(s(s(0)),w2))
neg(u(s(s(s(s(0)))),w2))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w2)))
neg(k2(s(s(s(0))),w2))
neg(u(s(s(s(s(s(0))))),w2))
neg(k2(s(0),b1))
neg(k1(s(s(s(s(s(s(s(0))))))),u(s(s(s(s(s(s(0)))))),w2)))
neg(k2(s(s(s(s(0)))),w2))
neg(u(s(s(s(s(s(s(0)))))),w2))
neg(b1)
neg(k1(s(s(s(s(s(s(s(s(0)))))))),u(s(s(s(s(s(s(s(0))))))),w2)))
w1
neg(k2(s(s(s(s(s(0))))),w2))
neg(u(s(s(s(s(s(s(s(0))))))),w2))
neg(k1(s(s(s(s(s(s(s(s(s(0))))))))),u(s(s(s(s(s(s(s(s(0)))))))),w2)))
yes

| ?-
```

C.3 Three-wise-men

The following is an actual scenario of queries. The query ``$step(i)$'' asks for all wffs which are true at step i. The predicates B_1, W_2, $s(i)$, $K_1(i, x)$, and $K_2(i, x)$ are represented as $b1$, $w2$, $s(i)$, $k1(i, x)$, and $k2(i, x)$, respectively. A universally quantified wff, $(\forall j)(\forall k)P(j, k)$, where $P(j, k)$ is some predicate expression, is represented as $forall([j, k], P(j, k))$. Valid formulas are built up from the predicates using neg, imp, and and in the expected way.

For ease of reading, wffs that were proven at the previous step (and subsequently inherited) have been removed from the subsequent step. All other inferred wffs are shown. Note that at step 17 the agent has indeed proved that his spot is white (i.e. W_1).

```
| ?- step(0).
yes

| ?- step(1).
forall([_63],k2(_63,forall([_64,_65,_66],imp(k3(_64,imp(_65,_66)),
  imp(k3(_64,_65),k3(s(_64),_66))))))
forall([_63],k2(_63,k3(s(0),imp(and(b1,b2),w3))))
forall([_63],k2(_63,imp(and(b1,b2),k3(s(0),and(b1,b2)))))
forall([_63],k2(_63,imp(neg(and(b1,b2)),imp(b1,w2))))
forall([_63],k2(_63,forall([_64],imp(neg(u(s(_64),w3)),
  neg(k3(_64,w3))))))
forall([_63,_64],imp(neg(k1(s(_63),u(_63,_64))),neg(u(_63,_64))))
forall([_63],imp(neg(u(_63,w3)),k2(s(_63),neg(u(_63,w3)))))
forall([_63,_64,_65],imp(k2(_63,imp(_64,_65)),imp(k2(_63,_64),
  k2(s(_63),_65))))
forall([_63,_64,_65,_66,_67],imp(and(k2(_63,imp(neg(and(_64,_65)),
  and(_66,_67))),k2(_63,neg(and(_64,_65)))),k2(s(_63),and(_66,_67))))
forall([_63,_64,_65,_66,_67],imp(and(k2(_63,forall([_68,_69,_70],
  imp(k3(_68,imp(_69,_70)),imp(k3(_68,_69),k3(s(_68),_70))))),
  k2(_63,k3(_64,imp(and(_65,_66),_67)))),k2(s(_63),imp(k3(_64,
  and(_65,_66)),k3(s(_64),_67)))))
forall([_63,_64],imp(and(k2(_63,forall([_65],imp(neg(u(s(_65),w3)),
  neg(k3(_65,w3))))),k2(_63,neg(u(s(_64),w3)))),k2(s(_63),
  neg(k3(_64,w3)))))
forall([_63,_64,_65],imp(and(k2(_63,imp(_64,_65)),k2(_63,neg(_65))),
  k2(s(_63),neg(_64))))
forall([_63,_64,_65,_66],imp(and(k2(_63,imp(and(_64,_65),_66)),
  k2(_63,neg(_66))),k2(s(_63),neg(and(_64,_65)))))
forall([_63],imp(b1,k2(_63,b1)))
imp(neg(b1),w1)
forall([_63],imp(neg(u(s(_63),w2)),neg(k2(_63,w2))))
yes

| ?- step(2).
...
k2(s(0),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(0),k3(s(0),imp(and(b1,b2),w3)))
k2(s(0),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(0),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(0),forall([_72],imp(neg(u(s(_72),w3)),neg(k3(_72,w3)))))
yes

| ?- step(3).
```

```
...
imp(k2(s(0),and(b1,b2)),k2(s(s(0)),k3(s(0),and(b1,b2))))
imp(k2(s(0),neg(and(b1,b2))),k2(s(s(0)),imp(b1,w2)))
k2(s(s(0)),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
neg(k1(s(s(0)),u(s(0),w2)))
neg(k1(s(s(0)),u(s(0),w3)))
k2(s(s(0)),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(0)),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(0)),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(0)),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(0)),forall([_72],imp(neg(u(s(_72),w3)),neg(k3(_72,w3)))))
yes

| ?- step(4).
...
neg(u(s(0),w2))
neg(u(s(0),w3))
imp(k2(s(s(0)),k3(s(0),and(b1,b2))),k2(s(s(s(0))),k3(s(s(0)),w3)))
imp(k2(s(s(0)),and(b1,b2)),k2(s(s(s(0))),k3(s(0),and(b1,b2))))
imp(k2(s(s(0)),neg(and(b1,b2))),k2(s(s(s(0))),imp(b1,w2)))
k2(s(s(s(0))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
neg(k1(s(s(s(0))),u(s(s(0)),w2)))
neg(k1(s(s(s(0))),u(s(s(0)),w3)))
k2(s(s(s(0))),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(0))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(0))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(0))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(0))),forall([_72],imp(neg(u(s(_72),w3)),neg(k3(_72,w3)))))
yes

| ?- step(5).
...
neg(u(s(s(0)),w2))
neg(u(s(s(0)),w3))
k2(s(s(0)),neg(u(s(0),w3)))
imp(k2(s(s(s(0))),k3(s(0),and(b1,b2))),k2(s(s(s(s(0)))),
  k3(s(s(0)),w3)))
imp(k2(s(s(s(0))),and(b1,b2)),k2(s(s(s(s(0)))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(0))),neg(and(b1,b2))),k2(s(s(s(s(0)))),imp(b1,w2)))
neg(k2(0,w2))
k2(s(s(s(s(0)))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w2)))
neg(k1(s(s(s(s(0)))),u(s(s(s(0))),w3)))
k2(s(s(s(s(0)))),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(0)))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(0)))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(0)))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(0)))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(6).
```

```
...
neg(u(s(s(s(0))),w2))
neg(u(s(s(s(0))),w3))
k2(s(s(s(0))),neg(u(s(s(0)),w3)))
imp(k2(s(s(s(s(0)))),k3(s(0),and(b1,b2))),k2(s(s(s(s(s(0))))),
  k3(s(s(0)),w3)))
imp(k2(s(s(s(s(0)))),and(b1,b2)),k2(s(s(s(s(s(0))))),k3(s(0),
  and(b1,b2))))
imp(k2(s(s(s(s(0)))),neg(and(b1,b2))),k2(s(s(s(s(s(0))))),
  imp(b1,w2)))
neg(k2(s(0),w2))
k2(s(s(s(s(s(0))))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(0))),neg(k3(0,w3)))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w2)))
neg(k1(s(s(s(s(s(0))))),u(s(s(s(s(0)))),w3)))
k2(s(s(s(s(s(0))))),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(0))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(0))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(0))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(0))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(7).
...
neg(u(s(s(s(s(0)))),w2))
neg(u(s(s(s(s(0)))),w3))
k2(s(s(s(s(0)))),neg(u(s(s(s(0))),w3)))
imp(k2(s(s(s(s(s(0))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(0)))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(0))))),and(b1,b2)),
  k2(s(s(s(s(s(s(0)))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(0))))),neg(and(b1,b2))),k2(s(s(s(s(s(s(0)))))),
  imp(b1,w2)))
neg(k2(s(s(0)),w2))
k2(s(s(s(s(s(s(0)))))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(s(0)))),neg(k3(s(0),w3)))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w2)))
neg(k1(s(s(s(s(s(s(0)))))),u(s(s(s(s(s(0))))),w3)))
k2(s(s(s(s(s(s(0)))))),forall([_72,_73,_74],imp(k3(_72,imp(_73,_74)),
  imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(0)))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(0)))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(0)))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(0)))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(8).
...
neg(u(s(s(s(s(s(0))))),w2))
neg(u(s(s(s(s(s(0))))),w3))
k2(s(s(s(s(s(0))))),neg(u(s(s(s(s(0)))),w3)))
imp(k2(s(s(s(s(s(s(0)))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(0))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(0)))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(0))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(0)))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(0))))))),imp(b1,w2)))
neg(k2(s(s(s(0))),w2))
k2(s(s(s(s(s(s(s(0))))))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(s(s(0))))),neg(k3(s(s(0)),w3)))
neg(k1(s(s(s(s(s(s(s(0))))))),u(s(s(s(s(s(s(0)))))),w2)))
neg(k1(s(s(s(s(s(s(s(0))))))),u(s(s(s(s(s(s(0)))))),w3)))
k2(s(s(s(s(s(s(s(0))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(0))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(0))))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(0))))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(0))))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(9).
...
neg(u(s(s(s(s(s(s(0)))))),w2))
neg(u(s(s(s(s(s(s(0)))))),w3))
k2(s(s(s(s(s(s(0)))))),neg(u(s(s(s(s(s(0))))),w3)))
imp(k2(s(s(s(s(s(s(s(0))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(0)))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(0))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(0)))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(0))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(0)))))))),imp(b1,w2)))
neg(k2(s(s(s(s(0)))),w2))
k2(s(s(s(s(s(s(s(s(0)))))))),imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(0)))))),neg(k3(s(s(s(0))),w3)))
k2(s(s(s(s(s(s(0)))))),neg(k3(s(0),and(b1,b2))))
neg(k1(s(s(s(s(s(s(s(s(0)))))))),u(s(s(s(s(s(s(s(0))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(0)))))))),u(s(s(s(s(s(s(s(0))))))),w3)))
k2(s(s(s(s(s(s(s(s(0)))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(s(0)))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(0)))))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(0)))))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(0)))))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(10).
...
neg(u(s(s(s(s(s(s(s(0))))))),w2))
neg(u(s(s(s(s(s(s(s(0))))))),w3))
k2(s(s(s(s(s(s(s(0))))))),neg(u(s(s(s(s(s(s(0)))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(0)))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(0))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(0)))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(0))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(0)))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(0)))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(0)))))),w2))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(0)))))))),neg(k3(s(s(s(s(0))))),w3)))
k2(s(s(s(s(s(s(s(0)))))))),neg(and(b1,b2)))
neg(k1(s(s(s(s(s(s(s(s(s(0)))))))))),u(s(s(s(s(s(s(s(s(0))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(0)))))))))),u(s(s(s(s(s(s(s(s(0))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes

| ?- step(11).
...
k2(s(s(s(s(s(s(s(s(0))))))))),imp(b1,w2))
neg(u(s(s(s(s(s(s(s(s(0))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(0))))))))),w3))
k2(s(s(s(s(s(s(s(s(0))))))))),neg(u(s(s(s(s(s(s(s(0)))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(0)))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(0)))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(0)))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(0))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(0))))))))),neg(k3(s(s(s(s(s(0)))))),w3)))
neg(k1(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  u(s(s(s(s(s(s(s(s(s(0)))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  u(s(s(s(s(s(s(s(s(s(0)))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),forall([_72],imp(neg(u(s(_72),w3)),
  neg(k3(_72,w3)))))
yes


| ?- step(12).
...
neg(u(s(s(s(s(s(s(s(s(s(0)))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(0)))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(0)))))))))),neg(u(s(s(s(s(s(s(s(s(0))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(0))))))))),b1),
  k2(s(s(s(s(s(s(s(s(s(0)))))))))),w2))
imp(k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),k3(s(s(0)),w3)))
```

```
imp(k2(s(s(s(s(s(s(s(s(s(s(0)))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(0)))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(0))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(0))))))))),neg(k3(s(s(s(s(s(s(0)))))),w3)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(0)))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(0)))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(and(b1,b2),k3(s(0),
  and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),forall([_72],
  imp(neg(u(s(_72),w3)),neg(k3(_72,w3)))))
yes

| ?- step(13).
...
neg(u(s(s(s(s(s(s(s(s(s(s(0)))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(s(0)))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(s(0)))))))))),
  neg(u(s(s(s(s(s(s(s(s(s(0))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(s(0)))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(s(0)))))))))),neg(k3(s(s(s(s(s(s(s(0))))))),w3)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),forall([_72,_73,_74],
  imp(k3(_72,imp(_73,_74)),imp(k3(_72,_73),k3(s(_72),_74)))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  forall([_72],imp(neg(u(s(_72),w3)),neg(k3(_72,w3)))))
yes

| ?- step(14).
...
```

```
neg(u(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  neg(u(s(s(s(s(s(s(s(s(s(s(0))))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(s(s(0)))))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),
  neg(k3(s(s(s(s(s(s(s(s(0))))))))),w3)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  forall([_156D,_1575,_157D],imp(k3(_156D,imp(_1575,_157D)),
  imp(k3(_156D,_1575),k3(s(_156D),_157D)))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  forall([_157D],imp(neg(u(s(_157D),w3)),neg(k3(_157D,w3)))))
yes

| ?- step(15).
...
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  neg(u(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(s(s(s(0))))))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),imp(k3(s(0),and(b1,b2)),
  k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),
  neg(k3(s(s(s(s(s(s(s(s(s(0)))))))))),w3)))
neg(k2(s(s(s(s(s(s(s(s(0))))))))),b1))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  forall([_1869,_1871,_1879],imp(k3(_1869,imp(_1871,_1879)),
  imp(k3(_1869,_1871),k3(s(_1869),_1879)))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  forall([_1879],imp(neg(u(s(_1879),w3)),neg(k3(_1879,w3)))))
yes

| ?- step(16).
...
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  neg(u(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(s(s(s(s(0))))))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),
  neg(k3(s(s(s(s(s(s(s(s(s(s(0)))))))))),w3)))
neg(b1)
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  forall([_1BA5,_1BAD,_1BB5],imp(k3(_1BA5,imp(_1BAD,_1BB5)),
  imp(k3(_1BA5,_1BAD),k3(s(_1BA5),_1BB5)))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  forall([_1BB5],imp(neg(u(s(_1BB5),w3)),neg(k3(_1BB5,w3)))))
yes

| ?- step(17).
...
w1
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w2))
neg(u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w3))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),
  neg(u(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  k3(s(0),and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  k3(s(s(0)),w3)))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),and(b1,b2)),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  k3(s(0),and(b1,b2))))
imp(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),neg(and(b1,b2))),
  k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),imp(b1,w2)))
neg(k2(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))),w2))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  imp(k3(s(0),and(b1,b2)),k3(s(s(0)),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))),
  neg(k3(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))),w3)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),w2)))
neg(k1(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  u(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  forall([_1DE1,_1DE9,_1DF1],imp(k3(_1DE1,imp(_1DE9,_1DF1)),
  imp(k3(_1DE1,_1DE9),k3(s(_1DE1),_1DF1)))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  k3(s(0),imp(and(b1,b2),w3)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  imp(and(b1,b2),k3(s(0),and(b1,b2))))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  imp(neg(and(b1,b2)),imp(b1,w2)))
k2(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0))))))))))))))))),
  forall([_1DF1],imp(neg(u(s(_1DF1),w3)),neg(k3(_1DF1,w3)))))
yes

| ?-
```

D Programs (PROLOG Code) used in Implementations

This appendix contains the programs for the various implementations of the step-logics.

D.1 SL^0

This is the PROLOG code that implements SL^0.

```
/* Any or all of state's arguments may be uninstantiated when
   called.  If N is uninstantiated, a positive integer will be
   generated before continuing. */
state(N,X) :-
   gen_if_needed(N),
   state1(N,X),
   asserta((state(N,X))).

/* state1 must be called with N instantiated. */
state1(N,X) :-
   feed(N,X).
state1(N,X) :-
   mp(N,X).

/* gen_if_needed generates a non-negative integer if N is
   uninstantiated; otherwise, it does nothing. */
gen_if_needed(N) :-
   ( (var(N), num(M), N is M - 1) ; (true) ).

/* feed is used to feed in tautologies.  It must be called with N
   instantiated. */
feed(N,X) :-
   formula(X,N),
   ax(X),
   asserta((feed(N,X))).
feed(N,X) :-
   N > 0 ,
   M is N - 1 ,
   feed(M,X).

/* mp must be called with N instantiated. */
mp(N,X) :-
   NN is N + 1,
   plus(JJ,_,NN),
   J is JJ - 1,
   state(J,imp(Y,X)),
   plus(KK,_,NN),
   K is KK - 1,
   state(K,Y),
   asserta((mp(N,X))).
```

```
/* ax checks that its argument is a tautology. */
ax(imp(X,imp(Y,X))).
ax(imp(imp(neg(X),neg(Y)),imp(imp(neg(X),Y),X))).
ax(imp(imp(X,imp(Y,Z)),imp(imp(X,Y),imp(X,Z)))).

/* formula generates formulas with 'weight' = N.  It must be
   called with N instantiated, and N > 0; otherwise it will
   fail. */
formula(p(M),N) :-
   N > 0,
   M is N - 1.
formula(imp(X,Y),N) :-
   N > 2,
   M is N - 1,
   plus(K,L,M),
   formula(X,K),
   formula(Y,L).
formula(neg(X),N) :-
   N > 1,
   M is N - 1,
   formula(X,M).

/* plus generates two positive integers, L, M, such that L + M = N.
   plus must be called with N instantiated, N > 1; otherwise it
   will fail. */
plus(L,M,N) :-
   num(L),
   ( (L >= N , ! , fail ) ; (M is N - L) ).

/* If num is called with an uninstantiated argument, N, it will
   generate a positive integer; otherwise it will fail if N is
   not a positive integer. */
num(1).
num(N) :-
   num(M),
   N is M + 1.
```

D.2 $SL_7(OBS_{B_3}, INF_B)$

This section contains the PROLOG code that was used to solve the *Brother problem* and generate the results in Section 6.2. The observation-function used here is OBS_{B_3}. The obvious changes to the code are needed to produce the observation-functions for the other two scenarios.

```
/* s finds a formula provable at step N.  If all formulas
   provable at step N have already been determined prior
   to entry into this rtn., then s must only search the
   'state' DB; otherwise, it must call 'state1'.
   (To do this, all formulas provable at step N-1 must have
   already been determined; otherwise, s returns error msg.:
   'can't do yet'.)  If state1(N,X) succeeds, then state(N,X)
   is added to the DB (if it's not already there).
   s must be called with N instantiated.  */
s(N,X) :-
   N >= 0, completed_step(N), !, state(N,X).
s(N,X) :-
```

```
   N >= 0,
   not(completed_step(N)),
   ((N > 0,         /* only continue if completed_step(N-1) */
     M is N - 1,
     ((completed_step(M));
      (write('cant do yet') , !, fail)));  /* ow, N = 0 */
    (true)),
   ((state1(N,X),
     not_yet_asserted(state(N,X)));  /* ow, can't find any more
                                       values for state1(N,X) --
                                       we must have them all. */
    (var(X),
     asserta((completed_step(N))),
     create_all_closed_sub_forms(N), /* creates all closed-sub-
                            formulas of thms. proven at step N */
     !, fail)).

/* state1 uses the various rules of inference of SL7 to find
   an X for which state1(N,X) succeeds.  state1 is called only
   when N is instantiated. */
state1(N,X) :-
   N > 0, inheritable(N,X).
state1(N,X) :-
   nowf(N,X).
state1(N,X) :-
   obs(N,X).
state1(N,X) :-
   N > 0, mp1(N,X).
state1(N,X) :-
   N > 0, mp2(N,X).
state1(N,X) :-
   N > 0, introspect(N,X).
state1(N,X) :-
   N > 0, contradiction(N,X).

/* nowf asserts now(N) at each step N, N > 0. */
nowf(N,now(N)) :-
   N > 0.

/* obs is the observation-function for this step-logic. */
obs(2,forall(N,imp(now(N),imp(neg(k(N-1,p(0))),neg(p(0)))))).
obs(2,imp(p(1),p(0))).
obs(2,p(1)).

/* mp1 determines whether anything is provable from the
   previous step using modus ponens.  It is called only
   when N is instantiated. */
mp1(N,X) :-
   M is N - 1, !,
   state(M,imp(Y,X)),
   state(M,Y).

/* mp2 determines whether anything is provable from the
   previous step using an extended version of modus ponens.
   It is called only when N is instantiated. */
```

```
mp2(N,neg(Y)) :-
   M is N - 1, !,
   state(M,forall(X,imp(now(X),imp(neg(k(X-1,Y)),neg(Y))))),
   state(M,now(K)),
   L is K - 1,
   state(M,neg(k(L,Y))).

/* introspect determines what things are NOT known at step
   N-1.  All closed-sub-formulas of step N-1
   *********** except those of the form 'k(_,_)' *********
   are possibilities.  Introspection can only be done when
   all formulas for step N-1 have been proven.
   introspect is called only when N is instantiated. */
introspect(N,neg(k(M,X))) :-
   M is N - 1,
   completed_step(M),  /* can only introspect if have full set
                          of thms from previous step. */
   !,
   closed_sub_form(M,X),   /* returns single wff of step M. */
   X \= k(_,_),      /* for now, can't introspect on k wffs. */
   not(state(M,X)).

/* contradiction determines whether a contradiction exists
   at the previous step. It is called only when N is
   instantiated. */
contradiction(N,contra(M,X,neg(X))) :-
   M is N - 1, !,
   state(M,neg(X)),
   state(M,X).

/* inheritable determines whether a formula is inheritable
   from step N-1 to step N.  All formulas X are inherited
   from step N-1 to N unless X = now(_) or if at step N-1
   it is known that a contradiction existed at step N-2
   between X and something.  This rtn. is called only when
   N is instantiated. */
inheritable(N,X) :-
   M is N - 1, L is M - 1, !,
   state(M,X),
   not(X = now(_)),
   not(state(M,contra(L,X,_))),
   not(state(M,contra(L,_,X))).

/* create_all_closed_sub_forms is used to create all the
   closed-sub-formulas of the thms that are proven in
   step N.  N is bound upon entry.  This procedure is only
   called if completed_step(N) succeeds. */
create_all_closed_sub_forms(N) :-
   ((state(N,X),
     cacsf(N,X),
     fail);
    (true)).

/* cacsf adds closed_sub_form(N,Y) to the DB for all Y which
```

```
   are closed-sub-formulas of the formula X (including X).
   If X is unbound, cacsf fails.  N is always bound on entry. */
cacsf(N,X) :-
   nonvar(X),
   cacsf1(N,X),
   assert_if_nec(closed_sub_form(N,X)),
   !.
cacsf1(N,imp(X,Y)) :-
   ((cacsf(N,X), !, cacsf(N,Y));
    (cacsf(N,Y), !, fail)).  /* want cacsf(N,Y) executed even
                                 if cacsf(N,X) failed. */
cacsf1(N,k(M,Y)) :-
   integer(M), !, cacsf(N,Y).
cacsf1(N,neg(X)) :-
   cacsf(N,X).
cacsf1(N,forall(_,X)) :-
   cacsf(N,X).
cacsf1(N,p(X)) :- nonvar(X).
cacsf1(N,now(X)) :- nonvar(X).

/* not_yet_asserted asserts X and succeeds, if X is not
   already true in the DB; otherwise it fails.
   If X is not bound upon entry, not_yet_asserted fails. */
not_yet_asserted(X) :-
   nonvar(X),
   ((X, !, fail);
    (assertz((X)))), !.

/* assert_if_nec asserts X, if X is not already true in the DB.
   If X is not bound upon entry, assert_if_nec fails. */
assert_if_nec(X) :-
   nonvar(X),
   ((X) ;
    (asserta((X)))), !.

/* step is used to find all thms. true at step N. */
step(N,H) :- s-out(N,H,RC), RC = 0, !.
step(N)   :- s-out(N,RC),   RC = 0, !.
stepno(N) :- s-no-out(N,RC), RC = 0, !.

/* s-out calls s. If s succeeds, then RC is set to 1;
   otherwise RC is set to 0. */
s-out(N,H,RC) :- ((s(N,X), write(H,state(N,X)), nl(H), RC is 1);
                  (RC is 0)).
s-out(N,RC)   :- ((s(N,X), write(X), nl, RC is 1);
                  (RC is 0)).
s-no-out(N,RC):- ((s(N,X), RC is 1);
                  (RC is 0)).
```

D.3 $SL_7(OBS_{W_2}, INF_{W_2})$

This is the PROLOG code used to implement the *Two-wise-men problem*. As noted in Section 7.2.4, wffs were stripped of any universal quantifiers. Thus, any wff with an apparent free variable actually has an implicit universal quantifier. This trick of stripping the quantifiers made the implementation details less cumbersome.

```
/* s finds a formula provable at step N.  If all formulas
   provable at step N have already been determined prior
   to entry into this rtn., then s must only search the
   'state' DB; otherwise, it must call 'state1'.
   (To do this, all formulas provable at step N-1 must have
   already been determined; otherwise, s returns error msg.:
   'can't do yet'.)  If state1(N,X) succeeds, then state(N,X)
   is added to the DB (if it's not already there).
   s must be called with N instantiated.  */
s(N,X) :-
   N >= 0, completed_step(N), !, state(N,X).
s(N,X) :-
   N >= 0,
   not(completed_step(N)),
   ((N > 0,       /* only continue if completed_step(N-1) */
     M is N - 1,
     ((completed_step(M));
      (write('cant do yet') , !, fail)));  /* ow, N = 0 */
     (true)),
   ((state1(N,X),
     not_yet_asserted(state(N,X)));  /* ow, can't find
                           any more values for state1(N,X)
                           --- we must have them all. */
    (var(X),
     asserta((completed_step(N))),
     !, fail)).

/* state1 uses the various rules of inference of SL7 to find
   an X for which state1(N,X) succeeds.  state1 is called
   only when N is instantiated. */
state1(N,X) :-
   N > 0, inheritable(N,X).
state1(N,X) :-
   obs(N,X).
state1(N,X) :-
   N > 0, mp1(N,X).
state1(N,X) :-
   N > 0, mp2(N,X).
state1(N,X) :-
   N > 0, mp1a(N,X).
state1(N,X) :-
   N > 0, introspect(N,X).

/* obs is the observation-function for this step-logic. */
obs(1,imp(k2(I,imp(X,Y)),imp(k2(I,X),k2(s(I),Y)))).
obs(1,k2(s(0),imp(b1,w2))).
obs(1,imp(b1,k2(s(0),b1))).
obs(1,imp(neg(b1),w1)).
obs(1,imp(neg(u(s(I),w2)),neg(k2(I,w2)))).
obs(1,imp(neg(k1(s(I),u(I,w2))),neg(u(I,w2)))).

/* mp1 determines whether anything is provable from the
   previous step using modus ponens.  It is called only
   when N is instantiated. */
```

```
mp1(N,X) :-
   M is N - 1, !,
   state(M,imp(Y,X)),
   state(M,Y).

/* mp2 determines whether anything is provable from the
   previous step using an extended version of modus ponens
   (where the antecedent must only unify, as opposed to being
   identical).  It is called only when N is instantiated. */

mp2(N,A) :-
   M is N - 1, !,
   state(M,imp(Y,X)),
   state(M,B),
   Y = B, A = X.

/* mp1a determines whether anything is provable from the
   previous step using the contra-positive of modus ponens.
   It is called only when N is instantiated. */
mp1a(N,neg(X)) :-
   M is N - 1, !,
   state(M,imp(X,Y)),
   state(M,neg(Y)).

/* introspect determines whether a given utterance was made
   at step N-1.  introspect is called only when N is
   instantiated. */
introspect(N,neg(k1(s(I),u(I,w2)))) :-
   N > 2,  M is N - 1, L is M - 1, x(L,I),  !,
   not(state(M,u(I,w2))).

/* inheritable inherits all formulas from step N-1 to step N.
   This rtn. is called only when N is instantiated. */
inheritable(N,X) :-
   M is N - 1, !,
   state(M,X).

/* not_yet_asserted asserts X and succeeds, if X is not
   already true in the DB; otherwise it fails.
   If X is not bound upon entry, not_yet_asserted fails. */
not_yet_asserted(X) :-
   nonvar(X),
   ((X, !, fail);
    (assertz((X)))), !.

/* x finds the successor form of the integer N.
   It is called only when N is instantiated. */
x(0,0).
x(N,s(I)) :-
   N > 0, M is N - 1, !,
   x(M,I).

/* step is used to find all thms. true at step N. */
step(N,H) :- s-out(N,H,RC), RC = 0, !.
step(N)   :- s-out(N,RC),   RC = 0, !.
```

```
stepno(N) :- s-no-out(N,RC), RC = 0, !.

/* s-out calls s. If s succeeds, then RC is set to 1;
   otherwise RC is set to 0. */
s-out(N,H,RC) :- ((s(N,X), write(H,state(N,X)), nl(H), RC is 1);
                  (RC is 0)).
s-out(N,RC)   :- ((s(N,X), write(X), nl, RC is 1);
                  (RC is 0)).
s-no-out(N,RC):- ((s(N,X), RC is 1);
                  (RC is 0)).

steps(N,M) :-
   create(H,'temp'),
   write(H,'***** start ***** '),
   write(H,N), write(H,' to '), write(H,M), nl(H), close(H),
   ctr_set(0,N),
   fail.
steps(_,M) :-
   repeat,
      ctr_inc(0,X),
      open(H,'temp',a),
      write(H,'Time at start of step is '),
      time(T), write(H,T), nl(H),
      step(X,H),
      write(H,'Time at end of step is   '),
      time(S), write(H,S), nl(H),
      close(H),
      write('Finished step '),write(X),nl,
   X == M,
   !.
```

D.4 $SL_7(OBS_{W_3}, INF_{W_3})$

This is the PROLOG code used to implement the *Three-wise-men problem*. Note that *two* rules were written to correspond to Rule 3, the extended rule of *modus ponens* (see page 29).

```
/* s finds a formula provable at step N.  If all formulas
   provable at step N have already been determined prior
   to entry into this rtn., then s must only search the
   'state' DB; otherwise, it must call 'state1'.
   (To do this, all formulas provable at step N-1 must have
   already been determined; otherwise, s returns error msg.:
   'can't do yet'.)  If state1(N,X) succeeds, then state(N,X)
   is added to the DB (if it's not already there).
   s must be called with N instantiated.  */
s(N,X) :-
   N >= 0, completed_step(N), !, state(N,X).
s(N,X) :-
   N >= 0,
   not(completed_step(N)),
   ((N > 0,       /* only continue if completed_step(N-1) */
     M is N - 1,
     ((completed_step(M));
      (write('cant do yet') , !, fail)));  /* ow, N = 0 */
    (true)),
   ((state1(N,X),
     not_yet_asserted(state(N,X)));  /* ow, can't find
                         any more values for state1(N,X)
                         --- we must have them all. */
    (var(X),
     asserta((completed_step(N))),
     !, fail)).

/* state1 uses the various rules of inference of SL7 to find
   an X for which state1(N,X) succeeds.  state1 is called
   only when N is instantiated. */
state1(N,X) :-
   N > 0, inheritable(N,X).
state1(N,X) :-
   obs(N,X).
state1(N,X) :-
   N > 0, mp1(N,X).
state1(N,X) :-
   N > 0, mp2(N,X).
state1(N,X) :-
   N > 0, mp1a(N,X).
state1(N,X) :-
   N > 0, mp2a(N,X).
state1(N,X) :-
   introspect(N,X).
state1(N,X) :-
   N > 0, instantiate(N,X).

/* obs is the observation-function for this step-logic. */
obs(1,forall([J],k2(J,forall([I,X,Y],
                             imp(k3(I,imp(X,Y)),
                                 imp(k3(I,X),k3(s(I),Y))))))).
obs(1,forall([J],k2(J,k3(s(0),imp(and(b1,b2),w3))))).
obs(1,forall([J],k2(J,imp(and(b1,b2),k3(s(0),and(b1,b2)))))).
obs(1,forall([J],k2(J,imp(neg(and(b1,b2)),imp(b1,w2))))).
obs(1,forall([J],k2(J,forall([I],imp(neg(u(s(I),w3)),
                                     neg(k3(I,w3))))))).
obs(1,forall([I,X],imp(neg(k1(s(I),u(I,X))),neg(u(I,X))))).
obs(1,forall([I],imp(neg(u(I,w3)),k2(s(I),neg(u(I,w3)))))).
obs(1,forall([I,X,Y],imp(k2(I,imp(X,Y)),
                         imp(k2(I,X),k2(s(I),Y))))).
obs(1,forall([I,X,X1,Y,Y1],
             imp(and(k2(I,imp(neg(and(X,X1)),and(Y,Y1))),
                     k2(I,neg(and(X,X1)))),
                 k2(s(I),and(Y,Y1))))).
obs(1,forall([J,K,Z,Z1,W],
             imp(and(k2(J,forall([I,X,Y],
                                 imp(k3(I,imp(X,Y)),
                                     imp(k3(I,X),k3(s(I),Y))))),
                     k2(J,k3(K,imp(and(Z,Z1),W)))),
                 k2(s(J),imp(k3(K,and(Z,Z1)),k3(s(K),W)))))).
obs(1,forall([J,K],imp(and(k2(J,forall([I],imp(neg(u(s(I),w3)),
                                               neg(k3(I,w3))))),
                           k2(J,neg(u(s(K),w3)))),
                       k2(s(J),neg(k3(K,w3)))))).
```

```
obs(1,forall([I,X,Y],imp(and(k2(I,imp(X,Y)),k2(I,neg(Y))),
                         k2(s(I),neg(X))))).
obs(1,forall([I,X,X1,Y],imp(and(k2(I,imp(and(X,X1),Y)),
                                         k2(I,neg(Y))),
                            k2(s(I),neg(and(X,X1)))))).
obs(1,forall([I],imp(b1,k2(I,b1)))).
obs(1,imp(neg(b1),w1)).
obs(1,forall([I],imp(neg(u(s(I),w2)),neg(k2(I,w2))))).
/* mp1 determines whether anything is provable from the
   previous step using modus ponens.  It is called only
   when N is instantiated. */
mp1(N,X) :-
   M is N - 1, !,
   state(M,imp(Y,X)),
   state(M,Y).

/* mp2 determines whether anything is provable from the
   previous step using an extended version of modus ponens
   (where the antecedent must only unify, as opposed to being
   identical).  It is called only when N is instantiated. */

mp2(N,X) :-
   M is N - 1,
   state(M,forall(_,imp(Y,X))),
   state(M,Y).
mp2(N,X) :-
   M is N - 1,
   state(M,forall(_,imp(and(Y,Z),X))),
   state(M,Y),
   state(M,Z).

/* mp1a determines whether anything is provable from the
   previous step using the contra-positive of modus ponens.
   It is called only when N is instantiated. */
mp1a(N,neg(X)) :-
   M is N - 1, !,
   state(M,imp(X,Y)),
   state(M,neg(Y)).

/* mp2a determines whether anything is provable from the
   previous step using the contra-positive of the extended
   version of modus ponens.  It is called only when N is
   instantiated. */
mp2a(N,neg(X)) :-
   M is N - 1, !,
   state(M,forall(_,imp(X,Y))),
   state(M,neg(Y)).

/* introspect determines whether a given utterance was made
   at step N-1.  introspect is called only when N is
   instantiated. */
introspect(N,neg(k1(s(I),u(I,X)))) :-
   N > 2,  M is N - 1, L is M - 1, x(L,I),  !,
   whospot(X),          /* 1st introspect on w2, then w3. */
   not(state(M,u(I,X))).
```

```
/* instantiate instantiates the time arg. of k2 predicates
   occurring at step N-1.  instantiate is called only when N
   is instantiated. */
instantiate(N,k2(I,X)) :-
   M is N - 1, !,
   state(M,forall([J],k2(J,X))),
   x(M,I).

/* inheritable inherits all formulas from step N-1 to step N.
   This rtn. is called only when N is instantiated. */
inheritable(N,X) :-
   M is N - 1, !,
   state(M,X).

/* whospot returns w2 on the first call, and w3 upon
   backtracking. */
whospot(w2).
whospot(w3).

/* not_yet_asserted asserts X and succeeds, if X is not
   already true in the DB; otherwise it fails.
   If X is not bound upon entry, not_yet_asserted fails. */
not_yet_asserted(state(N,X)) :-
   nonvar(X),
   ((state(N,X), !, fail) ;
    (assertz((state(N,X))))), !.

/* x finds the successor form of the integer N.
   It is called only when N is instantiated. */
x(0,0).
x(N,s(I)) :-
   N > 0, M is N - 1, !,
   x(M,I).

/* step is used to find all thms. true at step N. */
step(N,H) :- s-out(N,H,RC), RC = 0, !.
step(N)   :- s-out(N,RC),   RC = 0, !.
stepno(N) :- s-no-out(N,RC), RC = 0, !.

/* s-out calls s. If s succeeds, then RC is set to 1;
   otherwise RC is set to 0. */
s-out(N,H,RC) :- ((s(N,X), write(H,state(N,X)), nl(H), RC is 1);
                  (RC is 0)).
s-out(N,RC)   :- ((s(N,X), write(X), nl, RC is 1);
                  (RC is 0)).
s-no-out(N,RC):- ((s(N,X), RC is 1);
                  (RC is 0)).

steps(N,M) :-
   create(H,'temp'),
   write(H,'***** start ***** '),
   write(H,N), write(H,' to '), write(H,M), nl(H), close(H),
   ctr_set(0,N),
   fail.
```

```
steps(_,M) :-
   repeat,
      ctr_inc(0,X),
      open(H,’temp’,a),
      write(H,’Time at start of step is ’),
      time(T), write(H,T), nl(H),
      step(X,H),
      write(H,’Time at end of step is   ’),
      time(S), write(H,S), nl(H),
      close(H),
      write(’Finished step ’),write(X),nl,
   X == M,
   !.
```

Bibliography

[All84] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123--154, 1984.

[Ari84] Arity Corporation, Concord, Massachusetts. *Arity/Prolog: The Programming Environment*, 1984.

[BP83] J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, 1983.

[Che80] B. Chellas. *Modal Logic.* Cambridge University Press, 1980.

[CM84] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, second edition, 1984.

[CPr84] SRI International, Menlo Park, California. *C-Prolog User's Manual*, February 1984.

[deK86a] J. deKleer. An assumption-based TMS. *Artificial Intelligence*, 28:127--162, 1986.

[deK86b] J. deKleer. Extending the ATMS. *Artificial Intelligence*, 28:163--196, 1986.

[deK86c] J. deKleer. Problem solving with the ATMS. *Artificial Intelligence*, 28:197--224, 1986.

[DMP85] J. Drapkin, M. Miller, and D. Perlis. Real-time default reasoning, relevance, and memory models. Technical Report TR-85-35, Systems Research Center, University of Maryland, College Park, Maryland, 1985.

[DMP86a] J. Drapkin, M. Miller, and D. Perlis. A memory model for real-time commonsense reasoning. Technical Report TR-86-21, Systems Research Center, University of Maryland, College Park, Maryland, 1986.

[DMP86b] J. Drapkin, M. Miller, and D. Perlis. On default handling: Consistency before and after. Technical Report TR-86-20, Systems Research Center, University of Maryland, College Park, Maryland, 1986.

[Doy82] J. Doyle. Some theories of reasoned assumptions: An essay in rational psychology. Technical report, Department of Computer Science, Carnegie Mellon University, 1982.

[DP86a] J. Drapkin and D. Perlis. A preliminary excursion into step-logics. In *Proceedings SIGART International Symposium on Methodologies for Intelligent Systems*, pages 262--269. ACM, 1986. Knoxville, Tennessee.

[DP86b] J. Drapkin and D. Perlis. Step-logics: An alternative approach to limited reasoning. In *Proceedings of the European Conf. on Artificial Intelligence*, pages 160--163, 1986. Brighton, England.

[EDP88] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. Technical Report CS-TR-2016, Department of Computer Science, University of Maryland, College Park, Maryland, April 1988.

[FH88] R. Fagin and Y. Halpern, J. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39--76, 1988.

[Get63] E. Gettier. Is justified true belief knowledge? *Analysis*, 23:121--123, 1963.

[Gra78] J. Grant. Classifications for inconsistent theories. *Notre Dame Journal of Formal Logic*, 19(3), 1978.

[Haa85] A. Haas. Possible events, actual events, and robots. *Computational Intelligence*, 1(2):59--70, 1985.

[Haa86] A. Haas. A syntactic theory of belief and action. *Artificial Intelligence*, 28:245--292, 1986.

[Hen73] G. Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, 4:145--180, 1973.

[HM86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of the Fifth National Conference on Artificial Intelligence*. AAAI, 1986. Philadelphia, PA.

[HS86] J. Halpern and Y. Shoham. A propositional modal logic of time intervals. In *Proceedings of the Symposium on Logic in Computer Science*. IEEE, 1986. Boston, MA.

[KL87] S. Kraus and D. Lehmann. Knowledge, belief and time. Technical Report 87-4, Department of Computer Science, Hebrew University, Jerusalem 91904, Israel, April 1987.

[Kon83] K. Konolige. A deductive model of belief. In *Proceedings of the 8th Int'l Joint Conf. on Artificial Intelligence*, pages 377--381, 1983. Karlsruhe, West Germany.

[Kon84] K. Konolige. Belief and incompleteness. Technical Report 319, SRI International, 1984.

[Kon85a] K. Konolige. A computational theory of belief introspection. In *Proceedings of the 9th Int'l Joint Conf. on Artificial Intelligence*, pages 503--508, 1985. Los Angeles, CA.

[Kon85b] K. Konolige. Experimental robot psychology. Technical Report 363, SRI International, 1985.

[Kow79] R. Kowalski. *Logic for Problem Solving*. North Holland, 1979.

[Lak86] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In J. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 325--340. Morgan Kaufmann, 1986. Monterey, CA.

[Lev84] H. Levesque. A logic of implicit and explicit belief. In *Proceedings of the 3rd National Conf. on Artificial Intelligence*, pages 198--202, 1984. Austin, TX.

[Lif87] V. Lifschitz. Formal theories of action (preliminary report). In *Proceedings of the 10th Int'l Joint Conf. on Artificial Intelligence*, pages 966--972. Morgan Kaufmann, 1987. Milan, Italy.

[Mat49] B. Mates. Diodorean implication. *Philosophical Review*, 58:234--242, 1949.

[MB83] J. Malik and T. Binford. Reasoning in time and space. In *Proceedings of the 8th Int'l Joint Conf. on Artificial Intelligence*, pages 343--345, 1983. Karlsruhe, West Germany.

[McC78] J. McCarthy. Formalization of two puzzles involving knowledge. Unpublished note, Stanford University, 1978.

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

[McC86] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1):89--116, 1986.

[McD82] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101--155, 1982.

[MD80] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13(1,2):41--72, 1980.

[Men87] E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth, Belmont, CA, 3rd edition, 1987.

[Mil56] G. Miller. The magical number seven plus or minus two. *The Psychological Review*, 63:81--97, 1956.

[Moo83] R. Moore. Semantical considerations on nonmonotonic logic. In *Proceedings of the 8th Int'l Joint Conf. on Artificial Intelligence*, 1983. Karlsruhe, West Germany.

[Moo85] R. Moore. *Formal Theories of the Commonsense World*, chapter A Formal Theory of Knowledge and Action, pages 319--358. Ablex Publishing Company, 1985.

[MP85] J. Minker and D. Perlis. Computing protected circumscription. *Journal of Logic Programming*, 4:235--249, 1985.

[MS87] E. McKenzie and R. Snodgrass. Extending the relational algebra to support transaction time. In *Proceedings of the SIGMOD Conference*, pages 454--466. ACM, 1987. San Francisco, California.

[Nil83] N. Nilsson. Artificial intelligence prepares for 2001. *AI Magazine*, 4(4):7--14, 1983.

[Per81] D. Perlis. *Language, Computation, and Reality*. PhD thesis, Department of Computer Science, University of Rochester, Rochester, New York, 1981.

[Per84] D. Perlis. Non-monotonicity and real-time reasoning. In *Proceedings of the Workshop on Non-monotonic Reasoning*. AAAI, October 1984. New Paltz, New York.

[Per85] D. Perlis. Languages with self reference I: Foundations. *Artificial Intelligence*, 25:301--322, 1985.

[Per86] D. Perlis. On the consistency of commonsense reasoning. *Computational Intelligence*, 2:180--190, 1986.

[Per87] D. Perlis. Circumscribing with sets. *Artificial Intelligence*, 31:201--211, 1987.

[Per88a] D. Perlis. Autocircumscription. To appear, *Artificial Intelligence*, 1988.

[Per88b] D. Perlis. Languages with self reference II: Knowledge, belief, and modality. *Artificial Intelligence*, 34:179--212, 1988.

[PM86] D. Perlis and J. Minker. Completeness results for circumscription. *Artificial Intelligence*, 28(1):29--42, 1986.

[Pol87] J. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481--518, 1987.

[PR84] G. Priest and R. Routley. Introduction: Paraconsistent logics. *Studia logica*, 43:3--16, 1984.

[Pri67] A.N. Prior. *Past, Present and Future*. Clarendon Press, Oxford, 1967.

[Rei78] R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 55--76. Plenum, 1978.

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

[Sho88] Y. Shoham. *Reasoning About Change*. MIT Press, Cambridge, MA, 1988.

[Suc86] L. Suchman. Plans and situated actions: the problem of human-machine communication. Technical report, Xerox PARC, 1986.

[Var86] M. Vardi. On epistemic logic and logical omniscience. In J. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 293--305. Morgan Kaufmann, 1986. Monterey, CA.

[Won86] W. G. Wong. Prolog: A language for artificial intelligence. *PC Magazine*, 5(17):247--263, October 1986.