SRI International

A Deduction Model of Belief and its Logics

Technical Note 326

August 16, 1984

Kurt Konolige
Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

This research was made possible in part by a gift from the System Development Foundation. It was also supported in part by Contract N00014-80-C-0296 from the Office of Naval Research, and by Contract F49620-82-K-0031 from the Air Force Office of Scientific Research.

Abstract

Reasoning about the knowledge and beliefs of computer and human agents is assuming increasing importance in artificial intelligence systems for natural language understanding, planning, and knowledge representation. A natural model of belief for robot agents is the deduction model: an agent is represented as having an initial set of beliefs about the world in some internal language and a deduction process for deriving some (but not necessarily all) logical consequences of these beliefs. Because the deduction model is an explicitly computational model, it is possible to take into account limitations of an agent's resources when reasoning.

This thesis is an investigation of a Gentzen-type formalization of the deductive model of belief. Several original results are proved. Among these are soundness and completeness theorems for a deductive belief logic; a correspondence result that relates our deduction model to competing possible-world models; and a modal analog to Herbrand's Theorem for the belief logic. Specialized techniques for automatic deduction based on resolution are developed using this theorem.

Several other topics of knowledge and belief are explored in the thesis from the viewpoint of the deduction model, including

- A theory of introspection about self-beliefs
- a theory of circumscriptive ignorance, in which facts that an agent doesn't know are formalized by limiting or circumscribing the information available to him.

This report is a slightly revised version of a thesis submitted to the Department of Computer Science at Stanford University in June, 1984, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Acknowledgments

I could not have produced a thesis with as many mathematical symbols and theorems as this one without generous help from many people. This is the place to thank them.

First and foremost, my reading committee: John McCarthy, Nils Nilsson, and Solomon Feferman, who gave their time, expertise, and most important, signatures.

I am especially indebted to Nils, who encouraged me in the hard days of conception, and prodded me through the long days of completion. He generously shared his exceptional insight and experience in the field. You can't find a better reader.

The researchers at SRI: most of them had a hand in the thesis in one way or another—listening to talks, reading drafts, making comments (not all of them directed at the thesis, naturally). They made the dreary grind of thesis work more bearable.

SRI, you lovable bloat of an organization: for paying my salary.

And, of course, my Friend jj.

Contents

Illustrations

Tables

1. Introduction

Two artificially intelligent (AI) computer agents begin to play a game of chess, and the following conversation ensues:

S_1: *Do you know the rules of chess?*
S_2: *Yes.*
S_1: *Then you know whether White has a forced initial win or not.*
S_2: *Upon reflection, I realize that I must.*
S_1: *Then there is no reason to play.*
S_2: *No.*

Both agents are state-of-the-art constructions, incorporating the latest AI research in chess playing, natural-language understanding, planning, *etc.* But, because of the overwhelming combinatorics of the game, not even the fastest forseeable computers would be able to search the entire game tree of chess to find out whether White has a forced win. Why then do they come to such an odd conclusion about their own knowledge of chess?

The chess scenario is based not on computations of the game tree of chess, but on the agents' reasoning about their knowledge of the game. Simply put, they have an inaccurate model of their own capabilities. S_1 knows that chess is a finite game, and thus reasons that, in principle, knowing the rules of chess is all that is required to figure out whether White has a forced initial win. After learning that S_2 does indeed know the rules of chess, he comes to the erroneous conclusion that S_2 also knows this particular consequence of the rules. And S_2 himself, reflecting on his own knowledge in the same manner, arrives at the same conclusion, even though in actual fact he could never carry out the computations necessary to demonstrate it.

To reason about the beliefs of an intelligent agent (artificial or otherwise), an AI system like S_1 or S_2 must assume some model of that agent's beliefs about the world. By *model* we mean an abstract characterization of the actual object under consideration. A model is an abstraction because, for the sake of simplicity, it normally does not have all of the properties of the object it is modelling. The chess scenario above is an anecdotal example of the way models of belief that are too simplistic can lead to behavior that is less than intelligent in artificial agents. In this case, the model makes the assumption that agents can deduce all of the logical consequences of their beliefs; we call this property *consequential closure*, since it assumes that the beliefs of an agent are closed with respect to the derivation of consequences. It is clearly not warranted for either mechanical or human agents, because some derivations, although they are logically correct, may not be computationally feasible; this is, in fact, illustrated by the chess scenario.

In this thesis we will be concerned with developing an appropriate formal model of belief for robot planning and problem-solving systems, where the robots must represent and reason about states of the world and how their actions affect them. We argue that the two most important properties of such a model are:

- agents can draw conclusions from an initial set of beliefs, but
- they do not necessarily derive all the logically possible ones.

The first of these properties reflects the observation that robot agents can represent facts about the world, and perform inferences to reason about these facts and draw conclusions from them. Chess-playing agents, for example, can reason about the rules of chess to show that a certain sequence of moves is legal. However, robot agents are also computational devices, and, as such, they have limitations—restraints on the time and space they can allocate to perform inferences. Thus arises the second property: certain inferences may be logically possible, but they are infeasible from a computational point of view. This is the point of the chess anecdote.

The best current formal models of belief can capture the first property. These models are based on the idea that an agent's beliefs can be represented as a set of *possible worlds* compatible with those beliefs. The possible-world model is successful in addressing a number of representational issues concerning knowledge and belief, especially those concerning quantification into the context of a belief. It also has an elegant and concise axiomatization in terms of normal modal logics, which we discuss in Chapter 8. It has been the formal basis for a number of important theoretical studies in AI, as well as implemented systems in which the representation of belief was a critical factor.

However, a shortcoming of the possible-world model is that it is inconsistent with the second property. The notion of consequential closure is inherent in the analysis of belief as compatibility with a set of possible worlds, because all logical consequences of an agent's beliefs are also true of each compatible world. Thus, while possible-world models are good at predicting what consequences an agent could *possibly* derive from his beliefs, they are not capable of predicting what an agent *actually* believes, given that the agent may have resource or other limitations impeding the derivation of the consequences of his beliefs.

Lest the reader think that the second property is not an important one in modeling belief, we will briefly illustrate several types of derivational incompleteness which agents may manifest in their beliefs, and argue that reasoning about such incompleteness plays a critical role in intelligent behavior.

Resource Incompleteness. The chess scenario shows that an agent's ability to derive consequences of beliefs is limited by his computational resources. Although individual inferences about the game tree (what the consequences of legal moves are) can be simple, the number of them required to figure out whether a player has a forced initial win is astronomical and beyond the computational abilities of any agent. A suitable model of belief must be able to represent resource-limited situations, in which an agent possesses the inferential capability to derive some consequence of his beliefs, but simply does not have the resources to do so.

Fundamental Logical Incompleteness. Agents often seem unable to draw simple conclusions from their beliefs, even when given adequate computational resources. To take an example from high-school algebra: a student who is confronted with the equation $x + a = b$ and asked to solve for x will not be able to do so if he doesn't know the rule that subtracting equals from each side leaves the equation valid. It is not that the student lacks sufficient mental resources to solve this problem; rather, his rules of inference for dealing with equational theories are logically incomplete. To contrast this type of incompleteness with the resource-limited incompleteness described in the chess example, we call it *fundamental logical incompleteness.*

Fundamental logical incompleteness can arise when an agent must deal with an unfamiliar world for which he lacks adequate inferential capabilities. It is important in many areas: for example, one can imagine an AI expert system acting as a tutor, where it is crucial to characterize the inferential capabilities of the trainee. This suggests that a model of belief must be able to account for an agent's inability to derive some "easy" logical consequences of his beliefs, where by "easy" we mean that the agent has available sufficient computational resources to perform the derivation. This cannot be a characteristic of models of belief that use consequential closure to model an agent's reasoning about beliefs.

Relevance Incompleteness. Agents will typically have available many more beliefs than those they need consider in solving any given problem. For example, if asked whether Ronald Reagan were standing or sitting at this moment, most people would not hesitate (unless puzzled by the intent of the question) before responding "I don't know." It seems obvious that we do not generate a large number of deductions in trying to figure out whether Reagan's body position is a consequence of some beliefs we have; rather, we reach the conclusion that *no* beliefs are relevant to the question at hand. Of course, having the ability to limit the beliefs that are considered means that an agent can choose the wrong set, so that there may be some chain of inference that leads to the desired conclusion, but the beliefs necessary to generate it were not considered. Thus an agent's derivational process may be incomplete because of the beliefs he chooses to consider for a given problem; we call this type of incompleteness *relevance incompleteness.*

Modeling relevance incompleteness is an impossibility if it is assumed that the beliefs of agents are consequentially complete. An agent's beliefs cannot be partitioned into those that are either relevant or not to a given problem, and then the consequences of the relevant set deduced; rather, *all* the consequences of beliefs are believed.

1.1 The Deduction Model

In this thesis we introduce a new formal model of belief called the *deduction model.* The beliefs of an agent are described by postulating a *belief subsystem*: a set of statements in an internal language that represent the *base* beliefs of an agent, together with processes for deriving consequences of the base set and relating beliefs to other cognitive components of the agent. The internal language is a logical language, and the derivation rules are deduction rules—hence the term *deduction* model. Because the process of deriving the consequences of belief can be represented in this model, it is possible to take into account computational limitations of belief derivation.

As suggested by the chess scenario, the deduction model was developed in an effort to define accurate models of the beliefs of AI robot planning systems.

Because we designed and constructed them, we can actually look at the design and answer questions about their internal "mental" structures. In effect, we are using these planning systems as subjects in a discipline that might be appropriately termed *robot experimental psychology*. The key difference between this study and its human counterpart is that we actually have available the design, at a high level of abstraction, of the robot's internal information structures.

How well do AI planning systems conform to the deduction model? Such planning systems always have an internal language that represents states of the world, be the language frames or semantic nets or even a first-order logical language. For these systems, reasoning about the world is an inferential process—that is, they perform syntactic manipulations of the internal language to derive new facts from an original set of beliefs. Further, from a logical point of view such syntactic manipulations are often incomplete in exactly the sense we have claimed: there are simple deductions that are never performed, even with adequate space and time resources. The reasons for this can vary, but certainly an important one is the following. A complete set of deduction rules for an internal language with the expressive power of first-order logic is not decidable, so there is no computational procedure that is guaranteed to answer the question of whether a symbol string is a consequence of a set of beliefs in a finite amount of time. Even in those cases which are decidable, complete decision procedures often are too expensive computationally. So system-builders design belief deduction systems that are incomplete, but which are computationally efficient for a particular domain. Many systems, for example, employ limited reasoning about equality: from $A = B$ and $B = C$ they may infer that $A = C$, but from $f(x) = f(x-1) \times x$ they won't infer that $f(4) = 24$, even with the appropriate number-theoretic axioms.

An important class of AI deduction systems is *rule-based*. The basic idea is to embed domain-specific control information within the axioms themselves, rather than in a complex control strategy. This information tells how the axioms should be applied, so that a simple inference mechanism can be customized to perform efficiently in the intended domain. Resource-limited behavior is achieved by writing the axioms in such a way that only a small number of the logically possible inferences are actually performed, but these are the most important ones for the domain in question.

The deduction model was developed specifically with rule-based systems in mind. In our formal investigation, we employ a mathematical abstraction of belief subsystems, called *deduction structures*, which consist of a set of base sentences and a set of deduction rules. A deduction structure has a very simple control strategy: all its rules are applied whenever possible, generating every sanctioned inference. Technically, they are *closed* with respect to deductions. Being *deductively* closed is not the same as being *consequentially* closed if the deduction rules are not logically complete. By using very weak deduction rules that are efficient for a given domain, deduction structures exhibit resource-limited behavior, in a manner similar to rule-based systems. An important technique here is to carry along the cost of derivation as part of the information contained in a derived formula. The deduction rules can check this cost, and limit their applicability when it exceeds a certain bound. We use this technique, for example, in solving the Chess Problem in Section 6.1.

We might think of deduction structures as being an appropriate abstraction for a kind of *robot commonsense reasoning*: when a robot agent knows a set of facts about the world, there are a certain number of consequences that he should be able to deduce without, as it were, thinking hard. For instance, if the agent believes that the book is on the table in front of him, he should also be able to infer that it is less than 100 miles away, that it cannot be denser than lead, and so on. These are inferences that follow almost automatically from believing a certain fact.

It is important to note what representational problems of belief we are *not* addressing with the belief subsystem abstraction. Three of these that are critical areas for AI research are reasoning about uncertain information, belief revision in the face of contradictory information, and the integration of belief processing with other parts of an agent's cognitive makeup (desires, goals, plans, *etc.*).

Because it represents beliefs in a computational, symbol-processing paradigm, the deduction model is compatible with current philosophical theories of human cognitive states, for example, Lycan's [40] homuncular theory of belief, Fodor's [13] speculative psychology on the language of thought, and Moore and Hendrix's [53] account of belief sentence semantics. It should not be supposed, though, that the deduction model gives a completely accurate account of human belief. Our current understanding of human cognitive processes is not even remotely capable of providing formal theories that describe the intricacies of human behavior. In terms of belief, the phenomena of forgetting, reasoning based on uncertain information, and the exact relationship between long- and short-term memory are but a few of the problematic human behavioral features for which well-informed theories are lacking. The deduction model is explicitly *not* an attempt to provide such theories, and makes no pretense of being able to model behavior of this sort. However, the deduction belief model can capture what are perhaps the two most important aspects of human belief and commonsense reasoning: the fact that people can draw conclusions from their beliefs, and that they do not necessarily derive all the logically possible ones.

The importance of having a *formal* model of belief or other cognitive components is often underestimated. Formal models have the advantage of concreteness: it is possible to prove rigorously what properties the model has and what predictions it makes. By starting with a formal model, we can outline clearly how the model fits or does not fit its intended domain. Since all attempts at representing the world involve abstraction of greater or lesser degree, having the abstraction "on the table" and open to mathematical scrutiny seems to be the only way we can understand, in a precise way, the nature of the abstraction. In defining the deduction model, we are careful to note what assumptions are made in the interests of technical feasibility, and what consequences these have for its accuracy in the intended domain.

A formal model is also necessary if we want to be sure that the language we are using to talk about belief actually does describe beliefs. For example, several of the approaches to axiomatizing belief that we will criticize in Section 8 define a predicate B or *Bel* or *Believes*, such that the *intended* meaning of $Bel(a, p)$ is that agent a believes proposition p. However, there is no way to know whether or not the axioms that are given for the predicates actually describe their intended meaning; indeed, it is often not even clear what the intended meaning of the predicate *is*—exactly what aspects of our intuitive notion of belief are being described. And, as Montague [49] has shown, a straightforward translation of our intuitive notions of intensional concepts like belief into axiomatic logics of this sort can lead to outright inconsistencies, in which there are *no* models of the belief predicate. By contrast, in this paper we define a belief logic B that is provably sound and complete with respect to the deduction model: every theorem of B is true of the deduction model, and every true fact about the deduction model that can be stated in the language of B is provable. We are thus assured that the logic is actually about beliefs as portrayed by the model.

1.2 Scope of the Thesis

The concern with finding an appropriate abstraction of belief, and a representationally adequate language for talking about it, is an enterprise that McCarthy [45] has termed the *epistemological* part of building an intelligent system. A system that is epistemologically adequate has enough knowledge in principle to perform the task for which it was designed. The primary emphasis of this thesis is to develop an epistemologically adequate formal model of belief for robot planning systems. The first nine chapters are concerned with this topic: presenting the deduction model of belief, and defining several logics that are sound and complete with respect to it.

For a formalism to be useful in building a working AI system, it must, in addition, be *heuristically adequate*, that is, there must be some way of efficiently computing useful results from the formalism. As experience with AI systems has shown, a heuristically adequate system can usually be built only when the characteristics of a particular application domain are taken into account. As such, the derivation of heuristic adequacy is beyond the scope of this thesis. Nevertheless, in Chapters 10, 11, and 12 we present some proof methods for the belief logic that are intuitively appealing and amenable to mechanization. It is hoped that these proof methods form a foundation from which heuristically adequate systems for particular applications could be constructed.

In brief, the thesis is concerned with deriving an epistemologically adequate theory of belief for robot planners that can be refined to heuristic adequacy for particular application domains.

There are also several other important goals of the thesis. One is to present a theory of *ignorance*, or reasoning about non-knowledge, based on the idea that the number of facts an agent uses to reason about a proposition is limited. Another topic is the development of a theory of *belief introspection*, which posits that an agent calls on his belief subsystem in a recursive manner to introspect about the beliefs he has concerning his own beliefs.

A third goal is to explore proof methods that take seriously Weyhrauch's [66] concept of *semantic attachment*, a technique in which partial models of a language are realized as computations that can return the truthvalue of some statements of the language. In terms of the deduction model of belief, the idea is to determine the truth of statements about an agent's beliefs by actually using a deduction process to simulate the agent's own belief deduction (as opposed to reasoning about belief deduction in the logic). Although semantic attachment has many advocates, relatively little experience has been accumulated as to the practical benefits of incorporating it into mechanized proof procedures.

An interesting and important technical goal is the comparative study of two different types of languages for representing the deduction model. There are two different types of formal languages that have been used in the study of belief. In the so-called *syntactic* approach, the language is a first-order metalanguage that contains terms whose denotations are expressions in an object language. The object language serves as the internal language of the deduction model, and predicates in the metalanguage express facts about these sentences, such as their inclusion in the base set of sentences. Thus, in a syntactic language, belief is expressed as a relation between sentences in the internal language and the believer, a natural interpretation in terms of the deduction model.

The other approach has been to use a modal language, in which there is an operator on sentences of the language whose intended meaning is belief. Generally the chosen semantics of the modal language has been the possible-world semantics. The two methods have thus been hard to compare, since they have different semantic bases. It is not clear whether or not the languages say the same thing about belief, that is, whether there is a straightforward translation between the languages that respects truthvalues.

Both approaches have their own merits. The syntactic language is more expressive, in that it allows full quantification over sentences of the object language (the modal language just allows quantification over individuals). It also benefits from a large and well-known body of proof theory and techniques, such as Herbrand's Theorem, that make mechanization easier. On the other hand, the modal language is representationally more compact, and doesn't suffer from the proliferation of confusing terms referring to object language expressions that the syntactic approach is prone to. The deduction model admits both modal and syntactic representations; therefore, a comparative study of the two approaches is undertaken in this thesis, including the issues of their compatibility, expressiveness, and ease of mechanization.

There are several original contributions that this thesis makes to the field of AI. The first is the development of the deduction model, a formal model of belief for robot planning systems, that takes into account the resource limitations in the derivation of beliefs.

A second contribution is the development of interpretations for two quite different formal languages in terms of the deduction model. This allows a comparison of the languages in terms of their expressiveness and utility. To the author's knowledge, it is the first time a reconciliation between the modal and syntactic approach to representing facts about belief has been attempted. The major result here is a *correspondence property* for the deduction model: in the limit of logically complete belief deduction rules, the logic of the deduction model reduces to the standard logics for the possible-world model of belief. The deduction model thus dominates the possible-world model, in the sense that the latter is a special case of the former.

There are several results that are important for mechanization of the proof methods. The use of *semantic attachment* to simulate the belief deduction process directly within the logic is, as far as the author knows, novel to this thesis. The only similar development in proof theory is the tableau methods of Kripke [35] for modal logics; we compare these in Chapter 3. A second result is a version of Herbrand's Theorem for a modal belief logic. Herbrand's Theorem is particularly important for mechanization, since it relates satisfiability of a set of sentences to satisfiability of ground instances of those sentences; it is the basis of almost all attempts at mechanizing first-order logic. This result is particularly surprising for the modal language, since it has been sought after but unproved up to now (see, for instance, the negative results of Haspel [17]. Fariñas del Cerro [11] proves a Herbrand Theorem for modal logic, but the language is restricted so that no quantifiers appear in the scope of a modal operator).

The theory of introspection based on the concept of a recursive call to the belief subsystem appears to be a novel approach. Its advantages are that it is intuitively appealing and has a computational setting. The logics of introspection that are defined in Hintikka [21], Levesque [38] or Moore [52], for example, are simply descriptive: they posit that an agent's introspective reasoning should have certain properties, without ever showing how these properties could arise in a computational setting.

The theory of ignorance advanced in this thesis is a novel application of the idea of *circumscription* of McCarthy [47]. A circumscription schema is the basis of McCarthy's work on solving the qualification problem: trying to formalize what properties *do not* hold in a situation, given a formal description of those that do. The original work here involves applying the idea of circumscription to a set of beliefs to say what an agent *does not* believe. Instead of a circumscription schema, which is very hard to mechanize, circumscribing beliefs in the deduction model leads to a simple axiomatization. Previous work on ignorance (see, for example, Goad [14]) was based on the possible-world model, was not derived in terms of the intuitively appealing idea of circumscribing beliefs, and does not extend to the deduction model.

1.3 Outline of the Thesis

The plan of the thesis is as follows. In the next chapter we define the deduction model, and discuss its formal characteristics in some detail. By examining AI planning systems, we form an abstraction called a *belief subsystem* that represents the part of an agent's cognitive makeup responsible for beliefs. The concept of a *deduction structure* is then developed as a mathematical formalization of belief subsystems.

Chapter 3 introduces the modal logic family B as a means of stating facts about the deduction model. It is a weak representational language in the sense that it does not have any free variables inside the scope of modal operators (*i.e.*, no *quantifying-in*), but it does allow the developement of the logic in its simplest form. This and the previous chapter on the deduction model are the two most important chapters to read. Following these, the model theory of B is developed in Chapter 4, using the mathematical framework of deduction structures. We give soundness and completeness proofs for B here.

Several topics that are useful in applying B to problem domains are covered in Chapter 5, including a theory of ignorance, a formalization of common belief, and a simple theory of situations. Having developed this framework, we present three problems in the representation of beliefs in Chapter 6, and show how B can solve them.

Chapter 7 develops the theory of introspection from the point of view of an *introspective machine*, which is an agent's view of his own beliefs in a computational setting. Several families of logics, the BSn families, are defined to characterize introspective belief subsystems with different properties. At this point the formalism has been developed sufficiently to compare it to other approaches in Chapter 8, and to analyze the differences between syntactic and modal languages for representing belief models.

In Chapter 9 we continue to expand the representational power of the belief logic, by taking up the development of a family qB whose language allows quantifying-in constructs.

Having developed the analytic machinery of deductive belief logics, we turn our attention to practical mechanical theorem-proving methods. These methods are

1. A modified Davis-Putnam decision procedure for unquantified B.

2. Resolution methods for qB with function terms.
3. A rule-based system for qB with function terms.

In Chapter 10 we generalize the propositional decision procedure of Davis and Putnam [9] to the logic family B. As a practical application, we solve the hard form of the Wise Man Puzzle (*i.e.*, the proofs of ignorance are included) in Appendix A. To the author's knowledge, this is the first automatic solution to this puzzle.

In Chapter 11 we develop the necessary semantic machinery of semantic trees for qB, and prove an analog to Herbrand's Theorem of first-order logic. Building on these results, we present a complete resolution method for qB in Chapter 12. Finally, we develop incomplete resolution strategies that are computationally efficient, namely unit resolution and rule-based systems.

2. The Deduction Model of Belief

The strategy we pursue in constructing a model of belief is to examine the way that typical AI robot planning systems (STRIPS and NOAH [54], WARPLAN [65], KAMP [1], *etc.*) represent and reason about the world. This leads to the identification of an abstract *belief subsystem* as the internal structure responsible for the beliefs of these agents. The characteristics of belief subsystems can be summarized briefly:

1. A belief subsystem contains a list of sentences in some internal ("mental") language, the *base beliefs*;
2. Agents can infer consequences of their beliefs by syntactic manipulation of the sentences of the belief subsystem;
3. The derivation of consequences of beliefs is incomplete, because of limitations of the inference process.

Having identified a belief subsystem as that part of an agent responsible for beliefs, our next task is to define a formal mathematical structure that models it accurately. The decisions to be made here involve particular choices for modeling the various components of a belief subsystem: What does the internal language look like? What kind of inference process actually derives consequences of the base set? and so on. The formal mathematical object we construct according to these criteria is called a *deduction structure*. Its main components are a set of sentences in some logical language (corresponding to the base set of a belief subsystem) and a set of deduction rules (corresponding to the belief inference rules), which may be logically incomplete. Because we choose to model belief subsystems in terms of logical (but perhaps incomplete) deduction, we call it the *deduction model of belief.*

2.1 Planning and Beliefs: the Belief Subsystem Abstraction

A robot planning system must represent knowledge about the world in order to plan actions that affect the world. Of course it is not possible to represent all the complexity of the real world, so the planning system uses some abstraction of properties of the real world that are important for its task; *e.g.*, it might assume that there are objects that can be stacked on top of one another in simple ways (the *blocks world* domain). The state of the abstract world at any particular point in time has been called a *situation* in the AI literature.

In general, the planning system will have only incomplete knowledge of a situation. For instance, if it is equipped with visual sensors, it may be able to see only some of the objects in the world. What this means is that the system must represent and reason about partial descriptions of situations. The process of deriving beliefs is a *syntactic* operation that takes as input sentences of the formal language, and produces new sentences as output. Let us call any new sentences derived by inferences *derived sentences*, and the process of deriving them *belief derivation.*

It is helpful to view the representation and deduction of facts about the world as a separate subsystem within the planning system; we call it the *belief subsystem.* In its simplest, most abstract form, the belief subsystem comprises a list of sentences about a situation, together with a deduction process for deriving consequences of these sentences. It is integrated with other processes in the planning system, especially the *plan derivation process* that searches for sequences of actions to achieve a given goal. In a highly schematic form, Figure 2.1 sketches the belief subsystem and its interaction with other processes of the planning system. The belief system is composed of the base sentences, together with the belief derivation

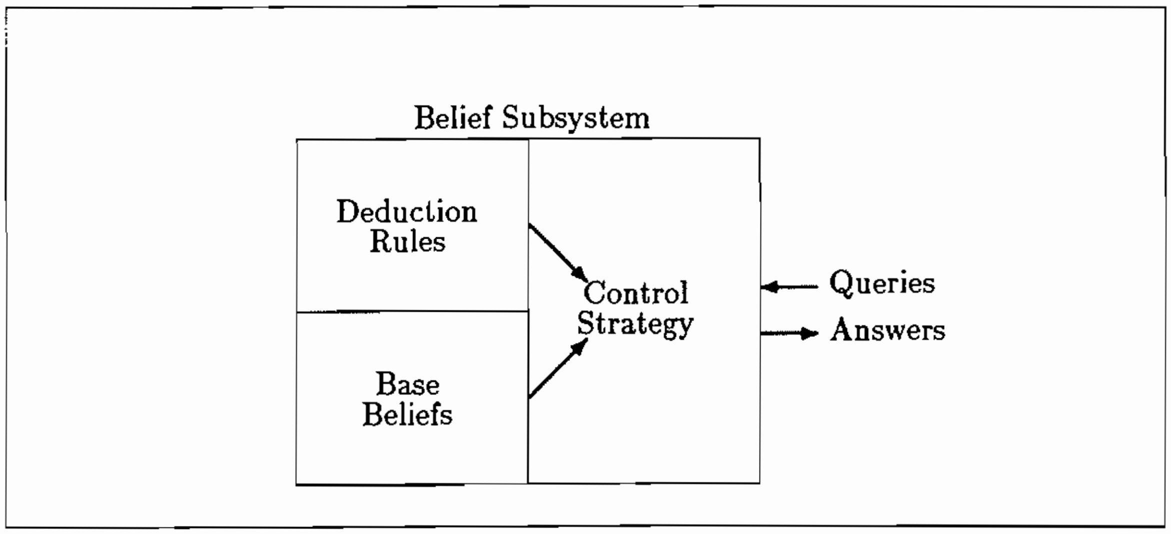

Figure 2.1 Schematic of a Belief Subsystem

process. Belief derivation itself can be decomposed into a set of deduction rules and a control strategy that determines how the deduction rules are to be applied and where their outputs go when requests are made to the belief subsystem.

There are two types of requests that result in some action in the belief subsystem. One is to add or delete sentences in the base set; this happens, for example, when the plan derivation process decides which sentences hold in a new situation. Belief updating and revision is a complicated research problem in its own right, and we do not address it here (see Doyle [10] for some related AI research). The second type of request is a query as to whether a sentence is a belief or not. This query causes the control strategy to try to prove, using the deduction rules, that the sentence is a consequence of the base set. It is this process of *belief querying* that we model in this paper. We now briefly examine some of its properties.

If we envision a belief subsystem as part of a robot agent that must interact with a changing environment, then the amount of time the agent can spend computing consequences of its beliefs is strictly limited: like human agents, robot agents will often have to act quickly to respond to a situation, without the luxury of unlimited resources for deriving a plan. Thus an important property of the belief query process is that it must always terminate in a finite (and usually small) amount of time.

What kinds of queries can be presented to the belief subsystem, and what is the nature of answers to these queries? There are two general forms of querying that have been used in AI systems, characterized by Chang and Lee [5] as Class A and B questions. Class A questions involve a simple *yes* or *no* answer, *e.g.*, "Is it raining outside?" For this type of question, a query in the form of a statement about the world is presented to the belief subsystem, and the subsystem tries to derive it on the basis of its base set. If it can, it answers *yes*; otherwise *no*.

Class B questions ask for information about individuals or conditions that satisfy an incomplete statement. For example, "Who knows whether it is raining or not?" is a Class B question, because it asks for an individual or set of individuals that satisfy a certain property. The answers to Class B questions can vary in their nature, depending on the particular circumstances involved. In some cases, an appropriate reply to the foregoing question is the name of an individual (*John*); in others, it might be a description of a single individual or class of individuals (*everyone who has been outside*), or even a disjunctive reply (*either John or Kim*).

Normally we will not be concerned with the distinction between these or any other types of queries that can be asked of a belief subsystem. The deduction model can accomodate any type of belief query as long as the process of belief derivation satisfies very general criteria, which are detailed in the next section. The only exception is with the specialized resolution proof methods which we develop in Chapter 12; to apply these techniques, the belief derivation process must be capable of answering a certain type of Class B question.

The foregoing description of the operation of a belief subsystem is meant to convey the idea that, in most formal planning systems, there is a tight interaction between belief subsystems and planning. Different systems may deviate from the described pattern to a greater or lesser extent. In some systems, the representation of facts may be so limited, and that of actions so explicit, as to almost obviate the need for belief deduction *per se* (as in some versions of STRIPS). In others, deduction may be used to calculate all the effects of an action by expanding the representation to include situations as objects (as in WARPLAN). It is hard to make a clean separation here between deductions performed for the purpose of deriving consequences of beliefs and those that establish the initial set of facts about a new situation. However, it is still conceptually useful to regard the belief subsystem as a separate structure and belief derivation as a separate process within the planning system.

2.2 A Formal Model of Belief

The formal mathematical object we use to model belief subsystems is called a *deduction structure*. It is a tuple consisting of two sets, and will be written as $\langle B, \mathcal{R} \rangle$. The set B is a set of sentences in some logical language L. It corresponds to the base set of a belief subsystem, and will be referred to as the *base sentences* of the deduction structure. About the only condition we require of L is that it be a *logical* language. Logical languages are distinguished by having a constructable set of syntactic objects, the *sentences* of the language, together with an *interpretation method* (a means of telling whether a sentence is true or false of a given state of affairs).

The set $\mathcal{R}$ is a set of deduction rules that operate on sentences of L. Its obvious correspondent in a belief subsystem is the rules of belief inference. We will leave unspecified the exact form of the deduction rules $\mathcal{R}$, but we do insist that they operate in the normal manner of deduction rules: there must be some method of applying the rules to derive conclusion sentences from premise sentences. If we think in terms of Hilbert systems (as defined in Kleene [28]), $\mathcal{R}$ would be a set of logical axioms (zero-premise rules) together with *modus ponens* (a two-premise rule). A sentence p would be derivable from the base sentences $B = \{b_1, b_2, \ldots\}$ if there were a Hilbert proof of $(b_1 \wedge b_2 \wedge \ldots) \supset p$, using the logical axioms and *modus ponens*. Sentences that can be derived from the base sentences are called *derivable sentences*.

A belief subsystem defines an agent's beliefs by the action of the deduction rules on the base set, under the guidance of the control strategy. Deduction structures model beliefs by defining a *belief set*, symbolized by $\mathrm{bel}(\langle B, \mathcal{R}\rangle, c)$, where c is a control strategy function. The belief set will contain the base sentences B, together with some of the derivable sentences. Just which derived sentences are included depends on how accurately we want to model the control strategy s of the corresponding belief subsystem. If the control strategy is intricate, the definition of the function bel could be complex, difficult to axiomatize, and computationally expensive to reason about. For this and several other reasons we make the important assumption that the belief set of a deduction structure is *closed under the deductions allowed by* $\mathcal{R}$: all derivable sentences are included in bel. One of the immediate technical gains of this assumption is that we can eliminate the need for a complex control strategy counterpart in deduction structures in favor of a simple closure condition on the belief set; hence we write $\mathrm{bel}(\langle B, \mathcal{R}\rangle)$, dropping the control strategy parameter. However, the correctness of the model may be impaired; we discuss this issue at some length below, and claim that deduction structures can still, in many cases, accurately capture the behavior of a belief subsystem's control strategy. It is important to note that deductive closure does not entail *consequential* closure: for example, a set of sentences closed under logically incomplete deduction rules need not contain all logical consequences of the set.

It is convenient to define an operator that symbolizes the process of belief derivation. We write $\Gamma \Vdash_{\mathcal{R}} p$ if p is derivable from the set of premises Γ using the rules $\mathcal{R}$. The belief set can then be defined as $\mathrm{bel}(\langle B, \mathcal{R}\rangle) = \{p | B \Vdash_{\mathcal{R}} p\}$. The belief derivation operator provides a concise method of summarizing various assumptions about belief derivation. The closure property, for example, can be stated as

$$\text{If } \Gamma \Vdash_{\rho(i)} p \ \text{ and } \ p, \Sigma \Vdash_{\rho(i)} q, \ \text{ then } \Gamma, \Sigma \Vdash_{\rho(i)} q,$$

where Γ and Σ are sets of sentences.

The belief derivation operator symbolizes a process, one that we assume we can actually run as a computation. The fact that belief subsystems always return an answer to a query in a finite amount of time implies that belief derivation is decidable, and thus that the process always terminates. We will not make this assumption in the thesis, but rather develop the logic of deduction structures in a more general framework in which belief derivation is only semidecidable, so that the process may not terminate for all inputs.

Finally, we single out certain sentences of the deduction structure for special treatment, namely those that themselves refer to the beliefs of agents. One of the key tests of a belief model is its ability to handle nested beliefs by assuming that agents use the model in representing other agents' beliefs; a belief model that has this characteristic is said to have the recursion property. In terms of deduction structures, the recursion property implies that the sentences of a deduction structure that are about beliefs should have another deduction structure as their intended interpretation.

At this point it is worthwhile to summarize the important properties of deduction structures and their associated belief sets.

LANGUAGE PROPERTY. The language of a deduction structure is logical language.

Deduction Property. The rules of a deduction structure are logical deduction rules. These rules are sound, effectively computable, and have a finite number of premises.

Closure Property. The *belief set* of a deduction structure is the least set that includes the base sentences and is closed under deductions.

Recursion Property. The intended model of sentences involving belief is the belief set of a deduction structure.

We now discuss each of these characteristics of the model in some detail. Readers who are not interested in the fine points of the model may wish to skip the rest of this chapter.

2.2.1 Language Property

About the only restriction we place on the language L of deduction structures is that sentences of the language have a well-defined model-theoretic semantics. This requirement seems absolutely necessary, for example, if we are going to talk about the beliefs of an agent being *true* of the actual world, or, as we will want to do in discussing the rationality of agents, judge the *soundness* of belief deduction rules. Such concepts make no sense in the absence of an interpretation method—a systematic way of interpreting the constructions of the language in terms of a model. Note that the interpretation method is not something that the agent carries around in his head; as far as the agent is concerned, a belief subsystem is just a collection of sentences, and computational processes manipulate the sentences themselves, and not their meanings. We simply cannot put the referent of "Cicero" inside our heads, even if he were alive. But the attribution of semantics to sentences is necessary if an outside observer is to analyze the nature of an agent's beliefs.

How well do actual robot belief subsystems fit in with the assumption of a logical language of belief? AI systems use a variety of representational technologies; chief among these are frames, scripts, semantic nets, and the many refinements of first-order logic (FOL), including PROLOG and the rule-based logics of μ-PLANNER, CONNIVER, QA4, and the like. The representations that fall into the latter category inherit their semantics from FOL, despite many differences in the syntactic form of their expressions. But what can we say about the first three? In surface form they certainly don't look anything like conventional mathematical logics; and their designers often have not provided anything but an informal idea of what the meanings of expressions in the language are. When, after all, is a pair of nodes connected by a directed arc *true* of the world? As Hayes [18] has forcefully argued, the lack of a model-theoretic semantics is a big drawback for these langauges. Fortunately, on further examination, it is often possible to provide such a semantics, usually by translating the representation into a first-order language. (See Woods [68] and Schubert [62] for a reconstruction of semantic nets in FOL terms, and Brachman [4] for a similar analysis of frames).

In the human sphere, at least one philosopher of mind has argued that internal representations that count as beliefs must have a model-theoretic semantics (see Fodor [13]). However, there almost certainly is a lot more to human belief than can be adequately handled within the framework of a logical language. For example, the question of membership in the belief set of a deduction structure is strictly bimodal: a sentence is either a member of the belief set of a deduction structure, or it isn't. If it is, then the assumed interpretation is that the agent believes that sentence to be true of the world. Deduction structures thus don't directly support the notion of uncertain beliefs, as they might if fuzzy or uncertain membership in the belief set were an inherent part of their structure.[1]

[1] However, uncertain beliefs could always be introduced into deduction structures in an *indirect* manner by letting L contain statements about uncertainty, *e.g.*, statements of the form P *is true with probability* 1/2.

It is often the case that we will want to fix the language of deduction structures in order to study their properties at a finer level of detail, *e.g.*, when looking at the behavior of nested beliefs in general, or when giving the particulars of the solution to a representational problem. It is convenient to think of the language as being a *parameter* of the formal model. For every logical language L, there are a set of deduction structures $\mathsf{D}(L)$ whose base sets are sentences of the language L.

2.2.2 Deduction Property

Rules for deduction structures are rules of inference with the following properties:

(Effectiveness)	The rule is an effectively computable function of sentences of L.
(Provinciality)	The number of premises is fixed and finite.
(Soundness)	The conclusion is sound with respect to the semantics of L.

These restrictions are those normally associated with deduction rules for classical logic, although strictly speaking deduction rules need not be sound, if one is interested strictly in the proof-theoretic properties of a logic, without regard to semantics.

The fact that belief deduction rules are general effectively computable functions means that they can be very complicated indeed. Mathematicians have been interested in logics with simple deduction rules (such as Hilbert systems) because it is easy to prove properties about the proof structure of such systems. However, for the purpose of deriving proof methods for common-sense reasoning in AI, it is often better to sacrifice simplicity for computational efficiency. Robinson's *resolution* rule [59], which employs a matching process called *unification*, is an example of a complicated rule that has found widespread employment in AI theorem-proving methods. Another important technique is Weyhrauch's *semantic attachment* [66], a general framework for viewing computation as deduction.

We call an inference rule *provincial* if the number of its input sentences is fixed and finite; deduction rules are always provincial. We thus do not allow inferences about beliefs that take an infinite number of premises. For example, Carnap's rule: *if for every individual a: $F(a)$ is a theorem, then $\forall x.F(x)$ is a theorem*, is not a valid rule of belief deduction.[2] Provincial inference rules have the following important property: if α is a consequence of a set of sentences S by the rule, then it is also a consequence of any larger set $S' \supset S$. To see that this must be so, consider that, if α can be derived by the application of provincial rules on the set of sentences S, and S' contains S, then the same derivation can be performed using S'. Rules that adhere to this property are called *monotonic*. Technically, monotonicity is convenient because it means we can reason about what an agent believes based on partial knowledge of his beliefs. A derivation made on the basis of a subset of his beliefs will always be valid, no matter what the full set of beliefs.[3]

Deduction rules for belief subsystems must also be sound. Soundness is a property of deductions defined in relation to the interpretations of the sentences they manipulate. A sound deduction rule is one for which, if the premises are true in an interpretation, then the conclusion will be also (see Kleene [28]). Informally, one would say that sound deduction rules never deduce false conclusions from true

[2] I am indebted to David Israel for pointing out this example.

[3] McCarthy (private communication) makes the point that there can be nonmonotonic rules that have a finite number of premises, if the language itself contains only a finite number of sentences. For example, let L contain the two predicates P and Q and the boolean opertors $\neg$ and $\vee$, but forbid all expressions with more than two boolean operators. There are only 38 different sentences in the language. We can define a nonmonotonic deduction rule *NM* on a set of sentences S as follows: return P if $\neg P$ is not among S. Because S is never larger than 38, the number of premises of *NM* is always less than 38. *NM* is nonmonotonic because if it is applied to an S not containing $\neg P$, it deduces P; but if $S' = \{\neg P\} \cup S$, *NM* will not deduce P.

The nonmonotonic nature of *NM* within a given deduction structure depends on restricting its application to the complete base set; for example, if it is applied to the subset S even when the base set is S', it will still deduce P. We argue that, because of this, *NM* is not a provincial deduction rule as we define it. A provincial deduction rule must have a fixed and finite set of premises; beyond that, there is no restriction on how the rule may be applied to a given base set of sentences. If it deduces p from a finite subset of sentences S of the base set, then it must always do so, no matter what other sentences the base set contains.

premises. *Modus ponens* is an example of a sound deduction rule: if p and $p \supset q$ are true, then q must also be true.

While the sound deduction rules of deduction structures can cover a large part of the process of belief derivation, there are other types of inferences that may be needed to form a useful model of belief. Some of these are

Belief revision:	the beliefs of an agent are updated to be consistent with new information (*e.g.*, Doyle [10]).
Default reasoning:	an agent "jumps to a conclusion" about the way the world is (*e.g.*, McCarthy [47], Reiter [58]).
Introspective reasoning:	an agent comes to a conclusion about the world based on his knowledge of his own beliefs (*e.g.*, Collins *et al.* [7], Moore [52]).

Of these three, we have already argued that the first, belief revision, is not a process that interacts with belief querying, and so we do not attempt to deal with it in the deduction model. However, the other two play a useful role in human belief inference, and deserve closer attention. We will call these two types of rules *extended inference rules.*[4] Let us examine some examples of extended inferences, and see why they cannot be couched directly in terms of deduction rules.

Suppose an agent knows that, in the typical case, birds have the ability to fly. A default rule of this sort might be informally stated as:

Rule F. If x is a bird, and nothing that is known about x contradicts it, assume that x can fly.

[4] There is a deliberate borrowing from Winograd's [67] terminology here. However, Winograd actually uses the term "extended inference modes," and makes it clear that he considers the defining quality of these inferences to be the lack of a standard semantics. By contrast, the extended inference rules we develop for circumscriptive ignorance and introspection in Chapter 7 have a perfectly well-defined semantics.

It is easy to show that Rule F is a nonmonotonic deduction rule. Suppose a base set S of sentences about Tweety is simply that Tweety is a bird; then by Rule F, it can be inferred that Tweety can fly. Now form the base set S' by adding the additional sentences that Tweety is an ostrich, and that ostriches can't fly. By simple deductions, the fact that Tweety can't fly can be derived; thus Rule F no longer applies, and even though $S \subset S'$, there are sentences that can be inferred from S that can't be inferred from S'.

As we have argued, all nonmonotonic inference rules must be nonprovincial. A default rule like Rule F specifies that something must be consistent with a set of sentences before it can be applied. This is what makes it *nonprovincial*: the premises of the inference rule are not a fixed, finite set. Default rules are also *unsound*, since there is an interpretation for S in which Tweety can't fly, yet, by default, it is assumed that he can. So default rules don't fit two restrictions that classical deduction rules obey.

A similar pattern emerges when we examine introspective inferences. These are conclusions that can be drawn on the basis of an agent's knowledge about his own beliefs. Consider the following line of reasoning by a feminist who has not studied much United States history:

> *Were there any female presidents? I certainly can't name all the presidents. On the other hand, I don't know of any female presidents, and if there had been any, I would have known it; therefore there mustn't have been any.*

This agent makes an inference based on the state of her own beliefs. The lack of a particular sentence in her belief subsystem leads to the derivation of another belief. Introspective reasoning of this sort has been studied by Collins *et al.* [7], who find it an important mode of reasoning for human subjects. It is obviously *nonprovincial*, because it depends on the state of the belief subsystem as a whole. On the other hand, as Moore [52] points out, introspective reasoning is distinguished from default reasoning in that it is *sound*: if the beliefs of an agent in its own information-gathering abilities are correct, then the inferences he makes about his lack of knowledge are perfectly justified.[5] By contrast, the use of a default rule always implies that an agent jumps to a conclusion that may not be correct.

The failure of extended inference rules to obey provinciality (and soundness) does not, by itself, preclude their use in deduction structures, although it does mean that their behavior would be different from standard deduction rules (*e.g.*, they would be nonmonotonic). However, whenever we try to add such rules to deduction structures, it turns out that they are also *not effectively computable* in the general case. That this should occur is not surprising. In the statement of typical default or introspective rules, there is a clause that refers to the state of an agent's beliefs; in Rule F above, this was the part that stated *and nothing that is known about x contradicts it ....* If we take a rule that refers to the process of belief derivation, like Rule F, and add it to the rules for a deduction structure, then the rule becomes self-referential, since it refers to the process of belief deduction in which it is embedded. As Reiter [58] has shown, the problem of deciding when it is permissible to apply a default rule of this sort is insoluble. This applies to systems like the default logic of Reiter [58] as well as the nonomonotic logics of McDermott and Doyle [48]. So we must agree with Israel [24] in his critique of nonomonotonic and default logics, that the intermixture of deduction and extended inference rules is ill-considered.

There is another way to view extended inference rules, namely, as operations on a *theory* considered as a syntactic whole. McCarthy [47] exploits this approach to formalize a certain type of useful default inference, which he calls *circumscription*. Because there is no self-reference involved in the circumscription rule (it does not use the theory it generates as input), it need not suffer from the problem of non-effectiveness inherent in the embedded logics discussed above.

[5] Moore [52] uses the term *autoepistemic reasoning* to indicate an inference made on the basis of self-knowledge.

Extended inference rules, when viewed as operations on theories as a whole, are compatible with the deduction model. In this approach, a deduction structure continues to function in the standard way, generating a belief set from the base sentences with deduction rules. Extended inference rules can use the deduction structure as input, for example to show that a certain statement is *not* in the belief set. In Chapter 5 we develop a theory of *circumscriptive ignorance* by defining an extended inference rule in analogy to McCarthy's rule; the theory of introspection of Chapter 7 also has extended inference rules for reasoning about self-beliefs.

To sum up: deduction structures are restricted to using deduction rules, which are provincial, sound, and effectively computable. Several interesting types of reasoning, such as reasoning about defaults or self-beliefs, cannot be modeled directly as deduction rules over sentences. However, they can be incorporated into the deduction model if the input to the rules is taken to be the deduction structure as a whole.

Closure Property. One of the key properties of belief subsystems that we wish to model is the incompleteness of deriving the consequences of the base set of beliefs. We have identified three sources of incompleteness in belief subsystems: an agent's belief inference rules may be too weak from a logical standpoint, or he may decide that some beliefs aren't relevant to a query, or his control strategy may perform only a subset of the derivations possible when confronted with resource limitations. All these methods are used by AI systems confronted with planning tasks under strict resource bounds. For several reasons, both conceptual and technical, we do not model incomplete control strategies directly in the deduction model. Instead, we make the assumption that the belief set of a deduction structure is *closed under derivations*: every sentence that can be derived from a set of sentences B by the rules $\mathcal{R}$ is in the belief set of $\langle B, \mathcal{R} \rangle$.

The closure property is an extremely important one, and we should examine its repercussions closely. An immediate point to make is that *derivational* closure is not the same as *consequential* closure. The latter refers to a property of sets of sentences based on their *semantics*: every logical consequence of the set is also a member of the set. The former refers to the *syntactic* process of derivability; and if the rules $\mathcal{R}$ are not logically complete, then a set of sentences that is derivationally closed under $\mathcal{R}$ need not be consequentially closed.

Probably the chief motivation for requiring derivational closure is that it simplifies the technical task of formalizing the deduction model. Consider the problem of formalizing a belief subsystem that has a complex control strategy guiding its derivations. To do this correctly, one must write axioms that describe the agendas, proof trees, and other data structures used by the control strategy, and how the control strategy guides deduction rules operating on these structures. Reasoning about the proof process involves making inferences using these axioms to *simulate* the process, a highly inefficient procedure. By contrast, the assumption of derivational closure leads to a simple formalization of belief subsystems that incorporates the belief deduction process in a direct way

Of course, it might be argued that, by assuming derivational closure, one cannot hope to model complex AI systems in which resource bounds play a central role in the proof process; for example, deduction structures would not be a correct formalization of planning systems that must operate under time constraints. However, on closer examination it can be shown that this criticism is almost entirely unfounded: the belief subsystems of present AI systems, even those with resource bounds, can in many instances be accurately modeled by deduction structures closed under derivations.

The first point to note in this regard is that many resource-limited proof processes, while they superficially seem to generate theories that are not closed with respect to derivations, are in fact isomorphic to a proof process that does. We need to make a distinction here between *local* and *global* effort bounds on a proof process. The distinction is quite simple. A local effort bound is a bound on the cost of a single derivation. For example, suppose the sentences A and B are in the the base set of a deduction structure, that C can be deduced from A and B by the rule R_1, and that D can be deduced from C by R_2. A derivation tree for D is:

$$\begin{tikzcd} A \arrow[dr, "R_1:"'] & & B \arrow[dl] \\ & C \arrow[d, "R_2:"'] & \\ & D & \end{tikzcd}$$

A local effort bound is a function on derivation trees. One of the simplest local bounds is to assign a cost to each derivation in a tree; if the cost of R_1 plus the cost of R_2 is greater than some threshold, the derivation tree above is not accepted as a proof of D.

By contrast, a global effort bound is a function over *all* the derivation trees produced by a proof process. Using a global bound, the derivation of D above might be accepted if it were produced early in the proof process, but not after other derivations have contributed to the global cost of the proof process.

One advantage of local over global effort bounds is conceptual clarity and predictability. Under a global bound, there is some control strategy that guides the proof process, making decisions to perform or not to perform deductions. The behavior of such a system is hard to predict. Theoretically there may be a derivation of a sentence, but the control strategy in a particular case decides not to derive it, because it tried other derivations first. Locally bounded systems, on the other hand, behave more dependably. They are guaranteed to arrive at all derivations that satisfy the local bound. Many theorem-proving strategies developed for AI actually use local bounds as a resource limitation; a good example is *level-saturation* as a resolution strategy (see Chang and Lee [5]).

Another important property of locally bounded proof processes is that they are always isomorphic to some derivationally complete proof process. We consider a simple example here. Suppose an agent uses *modus ponens* as his deduction rule, and has a control strategy in which only derivations using fewer than k applications of this rule are computed; this is a local effort bound. To model this situation with a derivationally closed deduction structure, consider transforming the base set so that each sentence has an extra conjunct tacked onto it, the predicate $DD(0)$ (DD stands for "derivation depth"). Instead of *modus ponens*, the deduction structure has the following modified rule:

$$MP2: \quad \frac{DD(n) \wedge \alpha \qquad DD(m) \wedge (\alpha \supset \beta)}{DD(n+m+1) \wedge \beta}, \quad n+m \leq k.$$

$MP2$ is sound and effectively computable, so it is a valid rule for a deduction structure. The closure of the base set of sentences of the structure under $MP2$ will be the same (modulo the DD predicate) as the set of sentences deduced by the nonclosed control strategy of the agent.

Finally, we note that many AI theorem-proving systems for commonsense reasoning embed their control knowledge in the sentences of the language, in a manner similar to that of rule $MP2$. These systems typically define new symbols for logical implication that have the same semantics as the normal implication sign ($\supset$), but are treated differently by the proof process. For example, one such sign ($P \rightarrow Q$) would be interpreted as meaning *if P is ever derived, then derive Q also.* It would never be used to infer $\neg P$ from $\neg Q$, although such an inference would be sound. By embedding control information in the sentences themselves, it is possible to use a uniform control strategy that is derivationally closed, where the proof process is guided by the syntactic form of the assertions made to the system. Systems of this sort, often called *rule-based* (see Nilsson [54]), make the same derivational closure assumption as deduction structures, and rely on the form of the axioms to achieve computational efficiency.

So, in retrospect, the decision to make belief sets closed under derivation is not a severe limitation on the accuracy of the Deduction Model. From a modeling viewpoint, the concept of "belief" is always going to be complicated by the introduction of control strategy issues. For example, it makes a difference to the control strategy as to whether a sentence is a member of the base set, or obtained at some point in a derivation. One cannot simply say, "Agent S believes P," because such a statement doesn't give enough information about P to be useful. If P is derived at the very limit of deductive resources, then nothing will follow from it; if it is a base sentence, then it might have significant consequences. By making the assumption of derivational closure, we have narrowed the scope of control strategies to those that use local effort bounds. This compromise allows us to arrive at a useful formalization of the deduction model, while still accurately portraying the most significant class of AI theorem-proving strategies for commonsense reasoning.

2.2.3 Recursion Property

There is one final topic to discuss about how deduction structures model belief subsystems: what predictions does the model make about *nested beliefs*, that is, beliefs about beliefs? If belief subsystems adhere to the recursion property, then agents view other agents as having belief subsystems similar to their own. This still leaves a large amount of flexibility in representing nested beliefs. For example, John might believe that Sue's internal language is L_1 and that she has a set of derivational rules $\mathcal{R}_1$, whereas Kim's internal language is L_2 and her derivational rules are $\mathcal{R}_2$. In addition, John might believe that Sue believes that Kim's internal language is L_3, and her rules are $\mathcal{R}_3$. We call the description of a belief subsystem at some level of nesting a *view*, and symbolize it with the Greek letter ν. Since the formal objects of the deduction model are deduction structures, these will be indexed by views when appropriate. For example, the view $\nu = John, Sue, Kim$ is Kim's belief subsystem as John believes that Sue sees it; a deduction structure formalizing this belief subsystem would be $d_{John,Sue,Kim}$. The actual world is represented by the empty view, $\nu = \emptyset$. Singleton views (the actual belief subsystems of each agent) can be indicated by using lowercase Roman indices, *e.g.*, $i = Kim$ is Kim's belief subsystem (the deduction structure d_{Kim}).

Obviously, some fairly complicated and confusing situations might be described, with agents believing that other agents have belief subsystems of varying capabilities. Some of these scenarios would be useful in representing situations that are of interest to AI systems, *e.g.*, an expert system tutoring a novice in some domain would need a representation of the novice's deductive capabilities that would initially be less powerful and complete than its own, and could be modified as the novice learned about the domain.

The recursion property does *not* imply that an agent's knowledge of another agent's beliefs is given by a deduction structure representing the latter's complete set of beliefs; rather, just as the outside observer uses a language to represent partial information about agents' beliefs, so an agent uses sentences of his internal language L to refer to the beliefs of agents. A standard construct is to have a *belief operator* in L: an operator whose arguments are an agent S and a sentence P, and whose intended meaning is that S believes P. The recursion property simply means that the belief operator must have a deduction structure as its interpretation. Deduction rules that apply to belief operators will be judged sound if they respect this interpretation. For example, suppose a deduction structure d has a rule stating that, from the premise sentences "John believes p" and "John believes $p \supset q$," the sentence "John believes q" can be concluded. This is a sound rule of d if *modus ponens* is believed to be a rule of the deduction structure modeling John's belief subsystem, since the presence of p and $p \supset q$ in a deduction structure with *modus ponens* means that q will be derived.

Several simplifying assumptions are implicit in the use of deduction structures to model the nested views of belief subsystems. The language L contains a belief operator that talks about membership in a belief set (its intended interpretation), and so L can describe what sentences are contained in an agent's belief set. However, there is no provision in L for talking about the deduction rules an agent uses. Instead, these nested belief rules are implicitly specified by the rules that manipulate sentences with belief operators. Consider the example from the previous paragraph. Let us suppose that we are modeling Sue's belief subsystem with the deduction structure d. Because Sue believes that John uses *modus ponens*, a sound rule of inference for d would be the one that was stated above, *viz.*, from the premise sentences "John believes p" and "John believes $p \supset q$" the sentence "John believes q" could be concluded. All of the rules that Sue believes John uses are modeled in this way. Similarly, if in Sue's opinion John believes that Kim uses a certain rule, then this will be reflected in a rule of John's deduction structure, which in turn will be modeled by a rule in d. The deduction model thus assumes that the rules for each view, though they may be different, are a fixed parameter of the model. We introduce the function $\rho(\nu)$ to specify deduction rule sets for each view ν; then for each function ρ and each language L, there is a set of deduction structures $\mathrm{D}(L, \rho)$ that formalize the deduction model.

The assumption of deduction rules as a fixed parameter of the model does not seem to be appropriate if we want to model changes in the rules as well as in the base set of beliefs. However, there is a way to overcome this difficulty. Just as we saw that we could model a control strategy that used local effort bounds by using more complicated deduction rules, we can model changing deduction rules by complicating the form of the internal language. Consider the high-school algebra example that was introduced on page 4. We have several choices of how we are to represent the *algebraic* rule: *subtracting equals from each side of an equation leaves the truth of the equation unchanged.* One way would be to have a deduction rule schema that says *from* $x = y$, *infer* $x - a = y - a$. Another equivalent implementation would be to have a general deduction rule such as *modus ponens*, and a proper axiom of the form $(x = y) \supset (x - a = y - a)$. By using proper axioms of this sort as belief sentences, we can model the changing state of an apprentice's knowledge of algebra. As we have seen, this is precisely the technique that AI rule-based systems exploit, embedding control information for deduction in the axioms themselves.

A final simplification, which is not inherent to the deduction model but which we introduce solely on the basis of technical convenience for this thesis, is that all deduction structures in all views use the same language L. There are situations where we might want to relax this restriction, but for the purposes of this thesis it makes the formalization of the deduction model less complex.

3. The Logic Family B

We now define a family of logics $\mathsf{B}(L, \rho)$ for stating facts and reasoning about deduction structures. This family is parameterized in the same way as deduction structures, namely by an agents' language L and an ensemble of deduction structure rules ρ. Each logic of the family is an axiomatization of the deduction structures $\mathsf{D}(L, \rho)$.

The axiomatization of B is complicated because it involves two languages. We have already discussed the agents' language L in the last chapter; it is used in defining deduction structures, and hence is a parameter of B. The language of B itself, which can be thought of as an outside observer's language (as opposed to the agents' langauge), is called L^{B}. It includes modal operators for stating that sentences are beliefs of an agent. Although in the most general case L and L^{B} will not be the same, it is often convenient to use L^{B} for the agents' language as well. We will normally make this assumption, but the major results of the thesis do not depend on it (for example, the soundness and completeness theorems in Chapter 4).

L^{B} does not contain any expressions denoting deduction rules of agents. Thus these rules are a parameter of the logic family, and are fixed once we decide to use a particular logic of the family. The ensemble function ρ picks out a set of rules for each agent. The reason we choose to make the deduction rules a parameter of B is that it is then possible to find efficient proof methods for B. One of the interesting features of B's axiomatization is that agents' rules are actually present as a subset of the rules of B; proofs about deduction structures in B use these rules directly in their derivation.

The logic of B is framed in terms of a modified form of Gentzen systems, the block tableau systems of Hintikka [20]. Although they may be unfamiliar to some readers, block tableaux are easy to work with and possess some natural advantages when applied to the formalization of deduction structures. Unlike Hilbert systems, which contain complex logical axioms and few but powerful rules of inference (*e.g.*, *modus ponens*), block tableau systems have simple axioms and a rich and flexible method of specifying deduction rules. We exploit this capability when we incorporate deduction structure rules into B.

In this chapter we first present a short overview of block tableaux. Then we give the postulates of the logic B, and present a particularization of B, **BK**, that uses a block tableau method for belief derivation. **BK** is the simplest possible system of this type, in the sense that it makes no assumptions about the knowledge an agent might have of his own beliefs (hence we call it *nonintrospective*). In Chapter 7 we introduce a theory of introspection based on belief subsystems, and define more complicated versions of B that capture various properties of introspection.

3.1 Block Tableaux

Most of this section will constitute a review for those readers who are already familiar with tableau systems.

3.1.1 The First-Order Language L_0

In what follows, we will use a standard first-order language, called L_0, as a base for defining other languages. It has constants but no other function symbols. In a later section on proof methods (Chapter 12) we will extend L_0 to functional terms; but these are not necessary for block tableaux systems, and complicate the exposition.

DEFINITION 3.1. *The first-order language L_0 is composed from the following symbols.*

1. *For each positive integer n, a denumerable set of* predicates of degree n *(generally roman capital letters, e.g., P, Q). We write P^n to indicate that P is of degree n.*
2. *A denumerable set of* individual variables *(usually small roman letters from the end of the alphabet, e.g., x, y).*
3. *A denumerable set of* individual constants *(usually small roman letters from the beginning of the alphabet, e.g., a, b).*
4. *The boolean connective symbols $\wedge$, $\vee$, $\supset$, and $\neg$.*
5. *The quantifier symbols $\exists$ and $\forall$.*

An *atomic formula* or *atom* of L_0 has the form $P^n(c_1, c_2 \ldots c_n)$, where each c_i is either an individual variable or constant. The normal formation rules for compound formulas of a first-order language are used.

We define substitution instances of quantified formulas.

DEFINITION 3.2. *Let α be a formula of L_0. For every variable x and individual constant a the formula α^x_a is given by the following set of inductive rules.*

1. *If α is atomic, then α^x_a is the result of substituting a for every occurrence of x in α.*
2. *If $\circ$ is a binary boolean operator, then*

$$\begin{aligned} (\alpha \circ \beta)^x_a &= \alpha^x_a \circ \beta^x_a \\ (\neg\alpha)^x_a &= \neg\alpha^x_a \quad . \end{aligned}$$

3. *If Q is a quantifier, then*

$$\begin{aligned} (Qx.\alpha)^x_a &= Qx.\,\alpha \\ (Qy.\alpha)^x_a &= Qy.\,\alpha^x_a \quad . \end{aligned}$$

A *closed formula* or *sentence* of L_0 is a formula with no free variables, i.e. $\alpha_a^x = \alpha$ for every x and a. A *ground atom* of L_0 is a closed atomic formula.

We will use lowercase Roman letters (p, q, *etc.*) as metavariables that stand for sentences of L_0, and lowercase Greek letters (α, ϕ, *etc.*) for formulas (which may also be sentences). Uppercase Greek letters ($\Gamma =_{\mathrm{df}} \{\gamma_1, \gamma_2, \ldots\}$, $\Delta =_{\mathrm{df}} \{\delta_1, \delta_2, \ldots\}$, *etc.*) stand for *finite sets* of sentences or formulas of L_0; in some few cases we let them stand for infinite sets, usually with a prime mark (Γ'). By ϕ, Γ we mean the set $\{\phi\} \cup \Gamma$. We also introduce the abbreviation $\neg\Gamma =_{\mathrm{df}} \{\neg\gamma_1, \neg\gamma_2, \ldots\}$.

An interpretation of L_0 is a truthvalue assignment to all sentences of L_0; this assignment must be a *first-order valuation*, that is, it must respect the standard interpretation of the Boolean connectives and the universal and existential quantifiers.

We call L_0 *uninterpreted* if every first-order valuation is an interpretation of L_0; *partially interpreted* if some subset of the first-order valuations are interpretations of L_0; and *fully interpreted* (or simply *interpreted*) if there is a singleton interpretation of L_0.

A sentence of L_0 is *valid* if and only if it is true in every interpretation of L_0.

3.1.2 Sequents

Sequents are the main formal object of block tableaux systems.

DEFINITION 3.3. *A sequent is an ordered pair of finite sets of sentences, $\langle\Gamma, \Delta\rangle$. This sequent will also be written as $\Gamma \Rightarrow \Delta$, and read as "$\Delta$ follows from Γ."*

A sequent $\Gamma \Rightarrow \Delta$ is true in an interpretation *of its component sentences iff one of γ_i is false, or one of δ_j is true. A sequent is* valid *iff it is true under all interpretations, and* satisfiable *iff it is true in at least one interpretation.*

From the definition of truth for a sequent, it should be clear that a sequent $\Gamma \Rightarrow \Delta$ is true in an interpretation just in case the sentence $(\gamma_1 \wedge \gamma_2 \wedge \ldots) \supset (\delta_1 \vee \delta_2 \vee \ldots)$ is true in that interpretation. Thus, in a given interpretation a true sequent can be taken as asserting that the conjunction of γ's materially implies the disjunction of the δ's.

We allow the empty set to appear on either side of a sequent, and abbreviate $\phi \Rightarrow \Delta$ by $\Rightarrow \Delta$, $\Gamma \Rightarrow \phi$ by $\Gamma \Rightarrow$, and $\phi \Rightarrow \phi$ by $\Rightarrow$. By the above definition, $\Rightarrow \Delta$ is true (in an interpretation) *iff* one of δ_i is true, $\Gamma \Rightarrow$ is true *iff* one of γ_i is false, and $\Rightarrow$ is never true in any interpretation.

In a few special cases, we allow denumerably infinite sets on either side of the sequent sign; the semantics is still given by Definition 3.3. Infinite sequents are never used in proof methods, but only to indicate the truthvalue of an infinite set of sentences.

3.1.3 Block Tableaux for L_0

The proof method that we adopt is similar to Gentzen's original sequent calculus, but simpler in form. It is called the *method of block tableaux*, and was originated by Hintikka [20]. A useful reference is Smullyan [63], in which many results in block tableaux and similar systems are presented in a unified form.

A block tableaux system consists of axioms and rules (collectively, *postulates*) whose formal objects are sequents. Block tableaux rules are like upside-down inference rules: the conclusion comes first, then a line, then the premises. Block tableaux themselves are derivations whose root is the sequent derived, whose branches are given by the rules, and whose leaves are axioms. Block tableaux look much like upside-down Gentzen system trees. (A more formal definition of a block tableaux is given below).

We consider a system $\mathcal{T}_0$ (see Smullyan [63], pages 105–109) that is first-order sound and complete: its consequences are precisely the sentences true in every first-order valuation.

DEFINITION 3.4. *The system $\mathcal{T}_0$ has the following postulates:*

Axioms. $\Gamma, p \Rightarrow \Delta, p$

Conjunction Rules.

$$C_1 : \frac{\Gamma, p \wedge q \Rightarrow \Delta}{\Gamma, p, q \Rightarrow \Delta}$$

$$C_2 : \frac{\Gamma \Rightarrow \Delta, p \wedge q}{\Gamma \Rightarrow \Delta, p \qquad \Gamma \Rightarrow \Delta, q}$$

Disjunction Rules.

$$D_1 : \frac{\Gamma \Rightarrow \Delta, p \vee q}{\Gamma \Rightarrow \Delta, p, q}$$

$$D_2 : \frac{\Gamma, p \vee q \Rightarrow \Delta}{\Gamma, p \Rightarrow \Delta \qquad \Gamma, q \Rightarrow \Delta}$$

Implication Rules.

$$I_1 : \frac{\Gamma \Rightarrow \Delta, p \supset q}{\Gamma, p \Rightarrow \Delta, q}$$

$$I_2 : \frac{\Gamma, p \supset q \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, p \qquad \Gamma, q \Rightarrow \Delta}$$

Negation Rules.

$$N_1 : \frac{\Gamma \Rightarrow \Delta, \neg p}{\Gamma, p \Rightarrow \Delta}$$

$$N_2 : \frac{\Gamma, \neg p \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, p}$$

Universal Rules.

$$U_1 : \frac{\Gamma, \forall x.\phi \Rightarrow \Delta}{\Gamma, \phi^x_a, \forall x.\phi \Rightarrow \Delta}$$

$$U_2 : \frac{\Gamma \Rightarrow \forall x.\phi, \Delta}{\Gamma \Rightarrow \phi^x_a, \forall x.\phi, \Delta},$$ *where a has not appeared in the tableau*

Existential Rules.

$$E_1 : \frac{\Gamma \Rightarrow \exists x.\phi, \Delta}{\Gamma \Rightarrow \phi^x_a, \exists x.\phi, \Delta}$$

$$E_2 : \quad \frac{\Gamma, \exists x.\phi \Rightarrow \Delta}{\Gamma, \phi^x_a, \exists x.\phi \Rightarrow \Delta}, \qquad \text{where } a \text{ has not appeared in the tableau}$$

Remarks. Note the simple form of the axioms and the symmetric nature of the inference rules (actually, each rule is a rule schema, since Γ, Δ, p, q, and ϕ stand for formulas and sets of formulas of L_0). There is one rule that deletes each logical connective on either side of the sequent. For example, the first conjunction rule deletes a conjunction on the left side of a sequent in favor of the two conjoined sentences; informally, it can be read as "Δ follows from Γ and $p \wedge q$ if it follows from Γ, p, and q." It is easily verified that each rule is *sound* with respect to first-order valuations: if the premises are true in an interpretation, then so is the conclusion. A proof of the completeness of $\mathcal{T}_0$ can be found in Smullyan [63].

DEFINITION 3.5. *A block tableau for the sequent $\Gamma \Rightarrow \Delta$ in a system $\mathcal{T}$ is a tree whose nodes are sequents, defined inductively as follows:*

1. *$\Gamma \Rightarrow \Delta$ is the root of the tree.*
2. *If sequent s is the parent node of daughters $s_1 \ldots s_n$, then $\dfrac{s}{s_1 \ldots s_n}$ is a rule of $\mathcal{T}$.*

A block tableau is closed *if all its leaves are axioms. If there is a closed block tableau for the sequent $\Gamma \Rightarrow \Delta$, then this sequent is a* theorem *of the system $\mathcal{T}$ and we write $\vdash_{\mathcal{T}} \Gamma \Rightarrow \Delta$.*

A system $\mathcal{T}'$ is called a subsystem of *$\mathcal{T}$ if every rule of $\mathcal{T}'$ is also a rule of $\mathcal{T}$; we write $\mathcal{T}' \sqsubseteq \mathcal{T}$. If some subsystem $\mathcal{T}'$ of $\mathcal{T}$ has exactly the same theorems as $\mathcal{T}$, then the rules of $\mathcal{T}$ not appearing in $\mathcal{T}'$ are said to be* eliminable from *$\mathcal{T}$, or* admissible to *$\mathcal{T}'$, and we write $\mathcal{T}' \simeq \mathcal{T}$.*

Block tableaux are similar to the AND/OR trees commonly encountered in AI theorem-proving systems (see Nilsson [54]). Rules such as C_2 cause AND-splitting, while a choice of rules to apply at a tableau node is an OR-split.

Example. Here is a block tableau for the sequent $\exists x.\, Bx \wedge Ax,\ \forall x.\, Cx \supset \neg Bx \Rightarrow \exists x.\, Ax \wedge \neg Cx$.

$$
\begin{array}{c}
\exists x.\, Bx \wedge Ax,\ \forall x.\, Cx \supset \neg Bx \Rightarrow \exists x.\, Ax \wedge \neg Cx \\
\hline
E_2 \quad Be \wedge Ae,\ \forall x.\, Cx \supset \neg Bx \Rightarrow \exists x.\, Ax \wedge \neg Cx \\
\hline
U_1 \quad Be \wedge Ae, Ce \supset \neg Be \Rightarrow \exists x.\, Ax \wedge \neg Cx \\
\hline
E_1 \quad Be \wedge Ae, Ce \supset \neg Be \Rightarrow Ae \wedge \neg Ce \\
\hline
C_1 \quad Ae, Be, Ce \supset \neg Be \Rightarrow Ae \wedge \neg Ce \\
\hline
I_2 \quad
\begin{array}{c}
Ae, Be, \neg Be \Rightarrow Ae \wedge \neg Ce \\
\hline
N_2 \quad Ae, Be \Rightarrow Be, Ae \wedge \neg Ce \\
\times
\end{array}
\qquad
\begin{array}{c}
Ae, Be \Rightarrow Ce, Ae \wedge \neg Ce \\
\hline
C_2 \quad
\begin{array}{c}
Ae, Be \Rightarrow Ce, \neg Ce \\
\hline
N_1 \quad Ae, Be, Ce \Rightarrow Ce \\
\times
\end{array}
\qquad
\begin{array}{c}
Ae, Be \Rightarrow Ce, Ae \\
\times
\end{array}
\end{array}
\end{array}
$$

The sequent to be proved is inserted as the root of the tree. By a series of reductions based on the rules of $\mathcal{T}_0$, the atoms of the sequent's sentences are extracted from the scope of quantifiers and Boolean operators. Splitting of the tree occurs with the rules I_2 and C_2; otherwise the reduction produces just a single sequent below the line. If a tree is found where the sequents at all the leaves are valid, then the theorem is proved, because it too must then be valid. Note that the logical inferences are from the leaves to the root of the tree, even though we work backwards in forming the tree. At each junction of the tree, the parent sequent is true in an interpretation if all its daughters are true in that interpretation.

An important connection between theoremhood and logical consequence for sequent systems is the following soundness theorem for tableaux.

THEOREM 3.1. *If $\Gamma \Rightarrow p$ is a theorem of $\mathcal{T}$ (where p is a single sentence of L_0), and all the rules of $\mathcal{T}$ are sound, then p is a logical consequence of Γ.*

Proof. If the rules of $\mathcal{T}$ are sound, then every theorem of $\mathcal{T}$ is valid. By Definition 3.3, this means that, in every interpretation in which all of Γ are true, p must be also.∎

3.2 The Language of B

There are two languages that concern the system B. First, there is the agents' language L, which we use to express the sentences that can be part of belief subsystems. L is a parameter of the system $\mathsf{B}(L, \rho)$. The second language is the one we use to express statements about the world and belief subsystems; this is the language *of* B, and is symbolized by L^{B}. L^{B} contains atoms of the form $[S_i]p$ to express belief, where S_i refers to an agent and p stands for an expression of the agents' language L.

In keeping with the requirements of the recursion property (see Section 2.2.3), we want L to also contain sentences that are about other agents' beliefs, *i.e.*, of the form $[S_i]p$. It is convenient, therefore, to consider systems in which the agents' language L and the language of the system itself (the outside observer's language) are one and the same: $L = L^{\mathsf{B}}$. We make this assumption henceforward for every logic of B, although it is a strictly a convenience.

The language L^{B} is built up from a first-order base language L_0 by the addition of the operators $[S_i]$ for belief. Under the assumption that $L = L^{\mathsf{B}}$, L can be determined once L_0 is given, and we will often write write $\mathsf{B}(L_0, \rho)$.

DEFINITION 3.6. *Let $S_1, S_2, \ldots$ be a countable sequence of (names for) agents. A* sentence *of L^{B} based on L_0 is defined inductively by the following rules.*

1. *All formation rules of L_0 are also formation rules of L^{B}.*
2. *If p is a sentence, then $[S_i]p$ is a sentence.*

An *ordinary atom* of L_0 is an atom of L^{B}; a *modal atom* is a sentence of the form $[S_i]p$. In the modal atom $[S_i]p$, p is said to be *in the context of* the modal operator. Note that there are no free variables in p, and hence no variables that are quantified outside of the contexts of a modal atom that appear inside its context; all modal atoms of $L^{\mathsf{B}}$are *ground*. The language L^{B} is extended in Chapter 9 to a language L^{qB} that includes quantification into modal contexts; logics based on L^{qB} have greater representational power (and also a more complex axiomatization) than those based on L^{B}.

The definition of a substitution instance of formulas of L_0 (Definition 3.2) is extended with the following rule for modal atoms.

4. $([S_i]\phi)^x_a = [S_i]\phi$

We will use the abbreviation $[S]\Gamma =_{df} \{[S]\gamma_1, [S]\gamma_2, \ldots\}$.

3.2.1 Interpretations

We give here an informal overview of the semantics of B to help the reader understand the calculus. The full model theory of B is presented in Chapter 4.

Interpretations of L^{B} are formed from interpretations of its base language L_0, together with an interpretation of modal atoms. Since modal atoms act like no-argument predications relative to the semantics of L_0, an interpretation of L^{B} is completely defined by an interpretation of L_0, together with an interpretation of the modal atoms: a truthvalue assignment that respects the intended meaning of the belief operator. In the deduction model, an interpretation of the modal atoms $[S_i]p$ is given by a deduction structure d_i. If p is in bel(d_i), then $[S_i]p$ is *true*; otherwise it is *false*.

Because L^{B} has no means of stating what the rules of a deduction structure are, we parameterize the family of logics $\mathsf{B}(L_0, \rho)$ in terms of a set of deduction rules $\rho(i)$ for each agent. These rules will be given beforehand for a particular logic of B and are fixed. The deduction structures corresponding to a logic $\mathsf{B}(L_0, \rho)$ are all in the class $\mathsf{D}(L_0, \rho)$.

The reader should note carefully that the semantics for B differs completely from that of most modal languages, in which the argument to the modal operator is usually taken to denote a *proposition* that can take on a truthvalue in a possible world. By contrast, arguments to modal operators in L^{B} denote a *sentence* of L_0, namely themselves. This distinction must be kept in mind when interpreting the modal operators of L^{B}.

3.3 A Sequent System for B

The deductive process that underlies the deduction model is characterized in very general terms by deduction structures and their associated belief sets. Until now we have been deliberately vague about the exact nature of deduction rules and the derivation process for an agent S_i, which is symbolized by $\Vdash_{\rho(i)}$. As stated in Chapter 2, there are only five conditions that must be satisfied: the deduction rules $\mathcal{R}$ must be *effective*, *provincial*, and *sound*, and the derivation operator $\Vdash$ *reflexive* and *closed under deduction*. Formally, we summarize these properties as follows:

(Effectiveness)	The deduction rules $\rho(i)$ are effectively applicable.
(Provinciality)	The number of premises of each rule is finite and bounded.
(Soundness)	If $\Gamma \Vdash_{\rho(i)} p$, then p is a logical consequence of Γ.
(Reflexivity)	$p \Vdash_{\rho(i)} p$.
(Closure)	If $\Gamma \Vdash_{\rho(i)} p$ and $p, \Sigma \Vdash_{\rho(i)} q$, then $\Gamma, \Sigma \Vdash_{\rho(i)} q$.

Suppose we are given beforehand a derivation operator $\Vdash_{\rho(i)}$ satisfying the above conditions for each agent S_i. The central problem in the formulation of $\mathsf{B}(L_0, \rho)$ is to find tableau rules that correctly implement the meaning of the belief operator $[S_i]$ under the derivations $\Vdash_{\rho(i)}$.

Consider first the sequent $[S_i]\Gamma \Rightarrow [S_i]p$. Its intended meaning is that, if all of Γ are in S_i's belief set, then so is p. The only possible way that we can guarantee this condition is if p is derivable from Γ for S_i, *i.e.*, $\Gamma \Vdash_{\rho(i)} p$. If this were not the case, then we could always construct the counterexample $d_i = \langle \Gamma, \rho(i) \rangle$ in which all of Γ are in d_i, but p is not. Thus we can relate the truth of a sequent involving belief operators to derivability in an agent's belief subsystem. This relation is captured by the inference rule

$$A: \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta}{\Gamma \Vdash_{\rho(i)} p} \quad .$$

A is called the *attachment rule*, because it derives results involving the belief operator by *attaching* sentences about belief to the derivation operator of an agent. Since premise is the bottom sequent and the conclusion the top, we can read A informally as follows: "If p is a deductive consequence of Γ in S_i's belief subsystem, then, whenever S_i believes Γ, he also believes p."

The attachment rule thus refers directly to the belief derivation operator. As discussed in Section 2.2, this operator stands for a derivation *process* that we actually have in hand. Given a set of agent's rules $\rho(i)$, we can run the belief derivation process to determine if p follows from Γ under $\rho(i)$, and hence whether $\Gamma \Vdash_{\rho(i)} p$ holds or not. Thus we can determine the validity of belief sentences in L^{B} by direct computation using an agent's belief derivtion process.

The rule A is not strictly an acceptable tableau rule, since in order to apply it we would have to extend the definition of block tableaux to include expressions based on the symbol $\Vdash$. A should really be defined as an axiom schema:

$$\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta \ , \quad \text{where } \Gamma \Vdash_{\rho(i)} p. \tag{3.1}$$

However, there are several advantages to retaining A in its rule form. The first advantage lies in the ability to incorporate different types of derivation processes into the sequent system. For example, in the next section we will particularize belief derivation to provability in a sequent system. Given this choice, the attachment rule will contain a sequent in the bottom half, and the steps of belief derivation become deduction steps of B.

A second advantage is that we need not worry about the question of the decidability of belief derivation. In the statement of the axiom, belief derivation must be decidable in order for (3.1) to be a valid axiom schema. But in the rule A, all that is required is that each of the steps in belief derivation be effective. We can define a branch containing an occurrence of A whose premise is $\Gamma \Vdash_{\rho(i)} p$ to be closed exactly when there is a proof of p from Γ using the rules $\rho(i)$. Each step in the construction of the tableau is thus effective, although theoremhood in B or in a belief subsystem may not be decidable.

The attachment rule is similar in many respects to a device originally proposed by Kripke [35] for the axiomatic study of modal systems. Kripke worked with *analytic tableaux*, a method similar to block tableaux. His idea was to introduce *auxiliary tableaux* as an addition to the main tableau of a proof. The purpose of the auxiliary tableaux is to show that a set of modal atoms of the form $\{[S_i]\Gamma, \neg[S_i]p\}$ is inconsistent, in much the same way that we show that $[S_i]p$ follows from $[S_i]\Gamma$ in the attachment rule. The analogy between the attachment rule and Kripke's technique is even more striking when we compare the system **BK** introduced in the next section, or the view windows of the resolution system **RK** in Section 12.2. Although the attachment rule was developed independently from Kripke's system, it can be considered to be a form of auxiliary tableaux adapted for the deduction model. .The chief difference lies in the identification of a separate set of deduction rules to model an agent's belief subsystem; auxiliary tableaux use the same rules as the main tableaux.

It is striking that two such widely different models, the deduction model and Kripke's possible-world model, led to an axiomatization that is so similar. We will explore this similarity in Chapter 8, and prove a correspondence property between the two models.

We now give an axiomatization for the logics of the family B.

DEFINITION 3.7. *The system* $\mathsf{B}(L_0, \rho)$ *has the following postulates.*

1. *The first-order complete rules* $\mathcal{T}_0$.
2. $A: \quad \dfrac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta}{\Gamma \Vdash_{\rho(i)} p}$
3. *A belief derivation operator* $\Vdash_{\rho(i)}$ *for each agent* S_i.

The logic family B is a compact formalization of the deduction model, and as such is useful in a theoretical analysis of the model, as in Chapter 4, where we prove the soundness and completeness of the axiomatization. It is also a starting point for finer-grained investigations of the nature of belief derivation. For instance, we might be interested in the behavior of subfamilies of B in which the rules of $\rho(i)$ that govern nested belief are as strong as A. In order to explore the fine structure of an agent's belief deduction process, we can fix the nature of $\Vdash_{\rho(i)}$ more precisely, and examine the resulting systems. These will all be particularizations of B in which the premise of the attachment rule A has been modified to correspond to the particular belief derivation method chosen.

3.4 The Nonintrospective Logic Family BK

The rich set of rules, and the flexibility of tableau derivations, make tableau systems a natural choice for the belief derivation process. In this section we define a particularization of B, the logic family BK, whose belief derivation process is defined in block tableaux terms.

DEFINITION 3.8. *Let* $d = \langle B, \mathcal{T}\rangle$. *A sentence* p *is* BK*-derivable from premises* Γ *in* d *if and only if* $\vdash_{\mathcal{T}} \Gamma, B \Rightarrow p$. *We will write this as* $\Gamma \Vdash_{\mathcal{T}} p$.

We need to show that tableau system derivability as just defined satisfies the five criteria of belief derivation: effectiveness, provinciality, soundness, reflexivity and closure. Consider a sequent system $\mathcal{T}$ made up of sound tableau rules. According to Theorem 3.1, the theorem $\vdash_{\mathcal{T}} \Gamma \Rightarrow p$ of $\mathcal{T}$ implies that p is a logical consequence of Γ, so we are assured that $\vdash_{\mathcal{T}}$ satisfies the soundness criterion. Provinciality and effectiveness are also satisfied, since the theorems of $\mathcal{T}$ are built by using effectively computable steps that operate on a finite number of sentences at each step. The observant reader might object at this point that tableau rules may indeed refer to an unbounded number of premise sentences; *e.g.*, any of the rules of $\mathcal{T}_0$ have this property, since Γ and Δ can stand for any set of sentences. However, each rule of $\mathcal{T}_0$ is actually a rule *schema*: the capital Greek letters are metavariables that are instantiated with a fixed, finite set of sentences to define a rule.

The closure condition is fulfilled by a special subclass of sequent systems, namely those for which the following rule, *Cut**, is admissible:

$$Cut^* : \quad \frac{\Gamma, \Sigma \Rightarrow p}{\Gamma \Rightarrow \beta \qquad \beta, \Sigma \Rightarrow p} \quad .$$

To see how this rule guarantees closure, suppose that $\Gamma \Rightarrow \beta$ and $\beta, \Sigma \Rightarrow p$ are both theorems of a sequent system $\mathcal{T}$ for which *Cut** is admissible. Because both premises of *Cut** have closed tableaux, the conclusion $\Gamma, \Sigma \Rightarrow p$ must also be a theorem.

Finally, the derivation process will be reflexive ($p \vdash_{\mathcal{T}} p$) if we include the following axiom in the system $\mathcal{T}$:

$$Id : \quad \Sigma, p \Rightarrow p, \Delta \quad .$$

Thus we only allow a system $\mathcal{T}$ to be used in belief derivation if the system is sound, *Cut** is an admissible rule of $\mathcal{T}$, and *Id* is an axiom of $\mathcal{T}$.

An interesting consequence of using tableau derivations in BK is that the attachment rule A can now be expressed wholly in terms of sequents, eliminating the derivation operator. To see how this comes about, consider first replacing the belief operator in rule A by tableau provability, as given by Definition 3.8. This yields

$$A'_K : \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta}{\vdash_{\tau(i)} \Gamma \Rightarrow p} \quad ,$$

where $\tau(i)$ is the set of tableau rules used by agent S_i.

Now $\vdash_{\tau(i)} \Gamma \Rightarrow p$ is true precisely if there is a closed tableau for $\Gamma \Rightarrow p$, using the rules $\tau(i)$. Hence we should be able to eliminate the provability symbol if we add the rules $\tau(i)$ to B for the purpose of constructing a tableau for $\Gamma \Rightarrow p$. In order to keep the agents' rules $\tau(i)$ from being confused with the rules of B, we add an agent index to sequent signs to indicate that the tableau rules are to be used for that particular agent only. The final version of the attachment rule is

$$A_K : \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta}{\Gamma \Rightarrow_i p} \quad .$$

Agents' rules are expressed using the indexed sequent sign, *e.g.*, if agent S_i were to use C_2, the following rule would be added to B:

$$C_2^i : \qquad \frac{\Gamma \Rightarrow_i \Delta, p \wedge q}{\Gamma \Rightarrow_i \Delta, p \qquad \Gamma \Rightarrow_i \Delta, q}$$

Taking the recursion property of belief subsystems seriously, we can iterate the process just described for the attachment rule. Each agent treats other agents as having a set of tableau rules. In formulating BK, there will be a tableau rule set associated with each *view* (views are discussed in relation to the recursion property in Section 2.2.3). We symbolize the set of tableau rules representing the view ν by $\tau(\nu)$.

A sequent $\Gamma \Rightarrow_\nu \Delta$, with index ν, is a statement about the belief subsystem of the view ν. For example, if $\nu = Sue, Kim$, the sequent $\Gamma \Rightarrow_\nu p$ states that p follows from Γ in Sue's view of Kim's belief subsystem. The deduction rules $\tau(\nu)$

always have sequents indexed by ν in their conclusions (above the line). This assures us that they will always be used as rules of the belief subsystem ν, and of no other.

The logic BK can thus be parameterized by a set of tableau rules for each view, and we write $\mathsf{BK}(L_0, \tau)$ to indicate this. If the sequent $\Gamma \Rightarrow_\nu \Delta$ is a theorem of the logic $\mathsf{BK}(L_0, \tau)$, it asserts that the sequent $\Gamma \Rightarrow \Delta$ is provable in the view ν; we write this as $\vdash_{\mathsf{BK}(L_0,\tau)} \Gamma \Rightarrow_\nu \Delta$. If this sequent is a theorem for every choice of L_0 and ρ, we write $\vdash_{\mathsf{BK}} \Gamma \Rightarrow_\nu \Delta$ or more simply $\vdash \Gamma \Rightarrow_\nu \Delta$ when BK is understood. Note that the presence of the index on the sequent means that we do not have to state explicitly that the set of rules used to derive the theorem were those of the view ν. Properties of the actual belief subsystems are always stated using an unindexed sequent, which we associate with the outside observer's view; for example, to show formally that if an agent believes p, then he believes q, we would have to prove that the sequent $[S_i]p \Rightarrow [S_i]q$ is a theorem of BK.

The exact form of A that is used in BK depends on the nature of the assumptions we wish to make about the introspective properties of belief—how an agent views his own belief subsystem. The simplest form of A is *nonintrospective*, in that it does not give any special consideration to an agent's view of his own belief subsystem. Note that this does not mean that we consider agents to have no knowledge of their own beliefs; just that there is no necessary connection between what they believe about their own beliefs and their actual beliefs. In Chapter 7 we develop a theory of belief introspection that addresses this issue in some detail; here we analyze the simpler system BK.

3.4.1 *Postulates of* $\mathsf{BK}(L_0, \tau)$

This family is parameterized by a base language L_0 and tableau rules $\tau(\nu)$ for each view ν. The belief derivation process is taken to be the method of block tableaux.

DEFINITION 3.9. *The system* $\mathsf{BK}(L_0, \tau)$ *is given by the following postulates:*

1. *The first-order complete rules* $\mathcal{T}_0$.
2. *The attachment rule*

$$A_K : \quad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]p, \Delta}{\Gamma \Rightarrow_i p} \quad .$$

3. *A set of sound sequent rules* $\tau(\nu)$ *for each view* ν *which contains the axiom Id, and for which the rule Cut* is admissible.*

Remarks. There are three parts to the system $\mathsf{BK}(L_0, \tau)$. The first part is a set of first-order rules for sentences about the actual world. These rules incorporate the nonsubscripted sequent sign ($\Rightarrow$), and are complete with respect to the semantics of first-order languages.

The second rule of $\mathsf{BK}(L_0, \tau)$ formalizes the way an outside observer views agents' belief subsystems. This is the sequent version of the attachment rule; it relates sentences involving the belief operator $[S_i]$ to the sequent system representing S_i's belief subsystem. As a simple example, let us assume that an agent S_i's belief subsystem is such that Q can be derived from P and $P \supset Q$. Then we can show that, if S_i believes P and $P \supset Q$, he believes Q:

$$A_K \quad \frac{[S_i]P, [S_i](P \supset Q) \Rightarrow [S_i]Q}{[S_i]P, [S_i](P \supset Q), P, P \supset Q \Rightarrow_i Q} \quad ,$$

which closes, since we assumed that $\vdash P, P \supset Q \Rightarrow_i Q$.

The rule A_K is a weak version of the attachment rule A in that it makes no assumptions about the beliefs an agent may have of his own beliefs. For example, we might argue that if an agent S believes a proposition P, he believes that he believes it. All he has to do to establish this is query his belief subsystem with the question, "Do I believe P?" If the answer comes back "yes," he should be able to infer that he does indeed believe P, *i.e.*, $[S][S]P$ is true if $[S]P$ is. However, as far as rule A_K is concerned, an agent's own belief subsystem has the same status for him as does that of any other agent. In particular, A_K allows an agent to have false and incomplete beliefs about his own beliefs.

The last part of BK is a set of rules formalizing the belief subsystem of each view. These rules involve the sequent sign $\Rightarrow_\nu$, since they talk about agents' deductive systems. They can contain rules that have a purely nonmodal import (*e.g.*, rules of $\mathcal{T}_0$), as well as rules that deal with belief operators. The rule *Cut**, which implements the closure property of belief sets, must be an admissible rule of $\tau(\nu)$, and the axiom *Id* must be present.

The rules $\tau(\nu)$ of a view ν can be incomplete in several ways. They may be first-order incomplete, in which case they cannot be used to draw all the consequences of sentences involving nonmodal operators that they otherwise might (to be first-order complete, it is sufficient for the rules $\mathcal{T}_0$ to be admissible in a view). Another type of incompleteness arises in reasoning about sentences involving belief operators. To be complete in this respect, a sufficient rule would be A_K. A view for which this rule is admissible is called *recursively complete.* If every view of a logic $\mathsf{BK}(L_0, \tau)$ is recursively and first-order complete, the logic is called *saturated.* We will symbolize the subfamily of saturated logics by BK_s.

3.4.2 *Some Theorems of* BK.

THEOREM 3.2. *Let p be derivable from Γ in the view i of* $\mathsf{BK}(L_0, \tau)$. *Then*

$$\vdash_{\mathsf{BK}(L_0,\tau)} [S_i]\Gamma \Rightarrow [S_i]p$$

Proof. In one step, using rule A_K:

$$A_K \quad \frac{[S_i]\Gamma \Rightarrow [S_i]p}{\Gamma \Rightarrow_i p} \quad \blacksquare$$
$$\times$$

THEOREM 3.3. *Let ν be a recursively complete view of* $\mathsf{BK}(L_0, \tau)$, *and let p be derivable from Γ in the view ν, i. Then*

$$\vdash [S_i]\Gamma \Rightarrow_\nu [S_i]p$$

Proof. In one step, using rule A_K of $\tau(\nu)$:

$$A_K \quad \frac{[S_i]\Gamma \Rightarrow_\nu [S_i]p}{\Gamma \Rightarrow_{\nu,i} p} \quad \blacksquare$$
$$\times$$

Remarks. These two theorems show that BK has a weakened analogue of the necessitation rule of modal logic (if p is provable, so is $\Box p$). If a nonmodal sentence p is provable in the view i (*i.e.*, $\vdash_{\mathsf{BK}(L_0,\tau)} \Rightarrow_i p$), then, by Theorem 3.2, $[S_i]p$ is provable in the empty view. Since the theorems of $\tau(i)$ are assumed to be sound, p is first-order valid, and so must be provable in the empty view.[6] Hence, for first-order sentences provable in the view i, necessitation holds. Theorem 3.3 establishes this result for an arbitrary view in which A is an admissible rule. Depending on the exact nature of the rule sets τ, necessitation will hold for some subset of the first-order valid sentences of a particular logic $\mathsf{BK}(L_0, \tau)$.

[6] Care must be taken in restricting p to nonmodal sentences, since the semantics of modal operators can change from one view to another (see the discussion of the recursion property in Section 2.2.3).

THEOREM 3.4. $\nvdash [S_i]p \Rightarrow p$

Proof. If p is a primitive sentence, then there is no applicable tableau rule, and hence no closed tableaux for the sequent. ∎

Remarks. The familiar modal logic principle $\Box p \supset p$ (if p is necessary, then p is true) is not a theorem of BK, since beliefs need not be true.

THEOREM 3.5. $\nvdash [S_i]p \Rightarrow [S_i][S_i]p$

Proof. The only applicable rule is A_K:

$$A_K \quad \frac{[S_i]p \Rightarrow [S_i][S_i]p}{p \Rightarrow_i [S_i]p}$$

According to the semantics of the deduction model, the sequent $p \Rightarrow_i [S_i]p$ is not valid: just because a sentence p is true does not mean that an agent S_i believes it. Hence, there cannot be any set of sound tableau rules for $\tau(i)$ that causes $p \Rightarrow_i [S_i]p$ to close. ∎

THEOREM 3.6. $\nvdash \neg[S_i]p \Rightarrow [S_i]\neg[S_i]p$

Proof. We can apply either N_2 or A_K. If we apply the latter, we obtain

$$A_K \quad \frac{\neg[S_i]p \Rightarrow [S_i]\neg[S_i]p}{\Rightarrow_i \neg[S_i]p}$$

But $\neg[S_i]p$ is not a valid sentence according to the deduction model, since it would require that no agent believe any sentence. Hence this tableau cannot close.

If we apply N_2 first, we obtain

$$N_2 \quad \frac{\neg[S_i]p \Rightarrow [S_i]\neg[S_i]p}{\Rightarrow [S_i]p, [S_i]\neg[S_i]p}$$

There are now two ways to apply A_K. In one application, we generate the sequent $\Rightarrow_i \neg[S_i]p$, which cannot close. In the other, we generate $\Rightarrow_i p$, which cannot be valid, since it would require agents to believe every sentence.∎

Remarks. These theorems show that no logic of **BK** sanctions any inferences about self-beliefs. If an agent believes a proposition p, it does not follow that his model of his own beliefs includes p; this is the import of Theorem 3.5. Similarly, if he does not believe a proposition, he also may not have knowledge of it; this is Theorem 3.6.

4. Model Theory for the Language B

In Chapter 3 we introduced the logic family B as a formal means of describing beliefs. With them, we proved some theorems about beliefs, *e.g.*, if P follows from Q in an agent's belief subsystem, then he believes P whenever he believes Q. These theorems are purportedly about beliefs as defined by belief subsystems: that is, our *intended* meaning for the statement $[S]p$ is that p is present in agent S's belief subsystem. However, we still have not shown that the calculi B are, in actual fact, *about* belief subsystems. To do this, we must show that the sentences of B can be interpreted in terms of deduction structures, the mathematical model of belief subsystems.

The study of the formal semantics of a language usually involves semantic-type formal objects: truthvalues, properties, domains of individuals, relations, possible worlds, and so on. An *interpretation* relates a structure composed of these objects, called a *model*, to the sentences of the language. For the purely nonmodal part of B, the model is standard: an interpretation of the base language L_0 is a first-order valuation of the sentences of L_0.

Modal atoms must also be assigned a truthvalue in an interpretation, and one that is in accord with their intended meaning as belief operators—the assignment must reflect the fact that they are sentences *about* belief subsystems, and not something else. For example, if $[S]p$ is true in an interpretation, and q follows from p for S, then $[S]q$ must also be true in that interpretation. In this scheme, a modal sentence $[S]p$ is assigned a value *true* if the sentence p is derived by the belief subsystem of agent S, and *false* otherwise. If we think of belief subsystems as being described by the belief set of a deduction structure, then the modal sentences $[S]p$ that are assigned a value *true* pick out the sentences p that are in S's belief set.

Thus the easiest and most convincing way to ensure that we have captured the intended meaning of the belief operator is to interpret modal sentences in relation to a set of deduction structures, one for each agent. In an interpretation, the belief set of an agent is modeled as the theory of a deduction structure: $[S]p$ is true in an interpretation exactly when p is a member of the theory of the deduction structure modeling agent S's belief subsystem.

There are several points to note about the formal model. The first is the presence of syntactic elements in the semantic domain. Deduction structures and theories consist of sets of sentences of L, and the proof-theoretic process of deduction is an integral part of the model. This seems to be a necessity if we want to be able to describe belief deduction as an incomplete process, *i.e.*, one that cannot be modeled by the complete logical consequences of a set of traditional semantic objects such as possible worlds.

A second observation is that a simple deduction structure model makes no claims about the interpretation of nested beliefs. Consider the sentence $[S_1][S_2]P$. This sentence is true in an interpretation if the sentence $[S_2]P$ is a member of the belief set of the deduction structure associated with agent S_1. Note that a question of truthvaluation has been converted into one of deduction. We are no longer asking what the *truthvalue* of $[S_2]P$ is, but rather if it can be *derived* in a certain system. How then can we guarantee that the atom $[S_2]P$ receives its proper interpretation as being *about* another belief subsystem, as required by the recursion property? The answer is that we demand that deduction structure rules be *sound* with respect to the intended meaning of the belief operator.

Soundness is a truth-theoretic property, and in order to talk about the soundness of deduction structure rules, we must assign a truthvalue to sentences of the internal language. The intended meaning of a nested belief like $[S_1][S_2]P$ must be formalized in terms of a deduction structure that represents S_1's view of S_2's belief subsystem, in accord with the recursion property discussed in Section 2.2.3. This is the course we pursue in developing a theory of belief introspection in Chapter 7.

In this chapter we introduce models for the calculi $\mathsf{B}(L, \rho)$, interpret the sentences of L^{B} with respect to these models, and show that the rules of B are indeed sound and complete with respect to this semantics. These models, called B-models, form the basis for all semantic studies of the deduction model. Recall that $\mathsf{B}(L, \rho)$ is parameterized by an agent's language L and a set of rules $\rho(i)$ for each agent S_i. B-models are also parameterized in this fashion—each agent S_i's belief subsystem is represented by a deduction structure whose base set comes from the language L^{B} and whose rules are $\rho(i)$ (the deduction structure class $\mathsf{D}(L, \rho(i))$). A $\mathsf{B}(L, \rho)$-model is composed of a set d_i of these deduction structures (one for each agent S_i), together with a first-order valuation of the nonmodal sentences of L^{B}. We will prove that $\mathsf{B}(L, \rho)$ is sound and complete with respect to its models. Note that we will generally distinguish between the agent's language L and the language L^{B} of B in this chapter, so that the results are valid even when these languages are not identical.

4.1 Models and Interpretations

As discussed, we define a class of models by fixing the language L and rules ρ of the deduction structures modeling each agent. The deduction structures are used to assign truthvalues to modal atoms, and a first-order valuation is used for nonmodal atoms. We need to introduce the machinery of first-order valuations here; we follow Smullyan [63] in this regard, since his method avoids some of the difficulties associated with scoping in quantified statements.

4.1.1 First-Order Valuations.

Consider a universe U of elements. We first define the notion of U-formulæ: formulæ of L^{B} with constants from U only.

DEFINITION 4.1. *An* atomic U-formula of L^{B} *is a tuple* $P\mathcal{E}_1 \ldots \mathcal{E}_n$, *where* P *is an* n*-ary predicate of* L^{B}, *and each* $\mathcal{E}_i$ *is either a variable or an element of* U. *A* U-formula of L^{B} *is a formula whose atoms are all atomic* U*-formulæ.* E^{U} *is the set of all closed* U*-formulæ.*

The set E^{U} includes all the *pure* sentences of L^{B}: those that do not have any constant terms or modal operators. We can define an interpretation for these sentences that respects the meaning of the quantifiers and Boolean operators; we call such an interpretation a *first-order valuation of* E^{U}.

DEFINITION 4.2. *A* first-order valuation v *of* E^{U} *is an assignment of truth values to all elements of* E^{U} *such that*

1. v *respects the meaning of the Boolean operators.*
2. $v(\forall x.\alpha) = \mathsf{t}$ *iff for every* $k \in \mathsf{U}$, $v(\alpha^x_k) = \mathsf{t}$.
3. $v(\exists x.\alpha) = \mathsf{t}$ *iff for some* $k \in \mathsf{U}$, $v(\alpha^x_k) = \mathsf{t}$.

An atomic valuation v_0 *is an assignment of* t *or* f *to every atomic element of* E^{U}.

Every atomic valuation of E^{U} can be extended to exactly one first-order valuation of E^{U} (this is proven informally in Smullyan [63]). Generally we will use atomic valuations to specify first-order valuations, since they are simpler.

The set E^{U} doesn't include any sentences with constant terms. To specify an interpretation for these, we use a mapping φ from constants to elements of U. Taken together, φ and v_0 specify a first-order valuation for every nonmodal sentence of L^{B}.

4.1.2 Hintikka Sets.

These sets are a useful tool for relating the semantics of sentences to tableau operations. Hintikka sets are formed from sentences of E^{U}.

DEFINITION 4.3. *A* Hintikka set *is a set of sentences W of L^{B} such that the following conditions hold:*

1. *No atomic element of E^{U} and its negation are both in W.*
2. *If $\neg\neg\alpha$ is in W, then α is in W.*
3. *If $\alpha \wedge \beta$ $(\neg(\alpha \vee \beta))$ is in W, then α $(\neg\alpha)$ and β $(\neg\beta)$ are in W.*
4. *If $\alpha \vee \beta$ $(\neg(\alpha \wedge \beta))$ is in W, then one of α $(\neg\alpha)$ or β $(\neg\beta)$ is in W.*
5. *If $\alpha \supset \beta$ $(\neg(\alpha \supset \beta))$ is in W, then one of $\neg\alpha$ (α) or β $(\neg\beta)$ is in W.*
6. *If $\forall x.\alpha(x)$ $(\neg\exists x.\alpha(x))$ is in W, then for* every *$k \in \mathsf{U}$, $\alpha(k) \in W$ $(\neg\alpha(k) \in W)$.*
7. *If $\exists x.\alpha(x)$ $(\neg\forall x.\alpha(x))$ is in W, then for* some *$k \in \mathsf{U}$, $\alpha(k) \in W$ $(\neg\alpha(k) \in W)$.*

An important property of every Hintikka set is that it is first-order satisfiable, *i.e.*, there exists a first-order valuation that satisfies every member of the set.

THEOREM 4.1. *Let U be the constants of L^{B}, and W a Hintikka set of sentences of L^{B}. There is a first-order valuation $\langle v_0, \varphi, \mathsf{U}\rangle$ of the sentences of W, where φ maps every constant to itself.*

Proof. The proof is in Smullyan [63].∎

4.1.3 $\mathsf{B}(L,\rho)$*-models.*

A $\mathsf{B}(L,\rho)$-model consists of a first-order valuation for L^{B} and a set of deduction structures from the classes $\mathsf{D}(L,\rho(i))$.

DEFINITION 4.4. *A* $\mathsf{B}(L,\rho)$-model *is a tuple $\langle v_0, \varphi, \mathsf{U}, D\rangle$, where v_0 is an atomic valuation of E^{U}, φ is a mapping from constants of L^{B} to* *elements of* U, *and* D *is a sequence consisting of one member from each of the classes* $\mathsf{D}(L, \rho(i))$.

We now define a valuation V of all sentences of L^{B} with respect to a $\mathsf{B}(L,\rho)$-model m.

DEFINITION 4.5. *Let* $m = (v_0, \varphi, \mathsf{U}, D)$ *be a* $\mathsf{B}(L,\rho)$*-model, and* s *a sentence of* L^{B}. *The valuation function* $V(s,m)$ *is defined by:*

1. $V(s,m)$ *is a first-order valuation that agrees with* v_0 *and* φ *when* s *is nonmodal.*
2. $V([S_i]p, m) = \mathsf{t}$ *iff* $p \in \mathrm{bel}(d_i)$, *where* $d_i \in D$.

The valuation function V is simply a first-order valuation that respects the meaning of the belief operator as indicating membership in the belief set of a deduction structure. A $\mathsf{B}(L,\rho)$*-interpretation* of the sentences of L^{B} is an assignment of truthvalues produced by the valuation function with respect to some model, *i.e.*, these interpretations are the class of functions defined by $I(s) =_{\mathrm{df}} \lambda s.V(s,m)$ for any $\mathsf{D}(L_0,\rho)$ model m. A sentence of L^{B} is $\mathsf{B}(L,\rho)$*-satisfiable* just in case it is true in some $\mathsf{B}(L,\rho)$-interpretation, and $\mathsf{B}(L,\rho)$*-valid* exactly when it is true in all $\mathsf{B}(L,\rho)$-interpretations. We will write $m \models s$ if $V(s,m) = \mathsf{t}$, and $\mathsf{B}(L,\rho) \models s$ if s is $\mathsf{B}(L,\rho)$-valid.

Example. Let L_0 contain the nilary predicate symbols P and Q. We wish to construct a model in which P is true (of the actual world), Q is false, the agent has the complete rules $\mathcal{T}_0$, and has P and $P \supset Q$ in his base set of beliefs (the universe U and mapping φ do not matter for this propositional language). A model that represents this situation is:

$$m =_{\mathrm{df}} (\{P\},\ \varphi,\ \mathsf{U},\ (\{P, P \supset Q\}, \mathcal{T}_0))$$

From m, we can compute the following truthvalues for sentences of L^{B}:

$$\begin{aligned}
V(P, m) &= \mathsf{t} \\
V(Q, m) &= \mathsf{f} \\
V(P \supset Q, m) &= \mathsf{f} \\
V([S]P, m) &= \mathsf{t} \\
V([S](P \supset Q), m) &= \mathsf{t} \\
V([S]Q, m) &= \mathsf{t}
\end{aligned}$$

The agent S has a false atomic belief about the world, namely Q.

4.2 Soundness and Completeness

We now prove the soundness and completeness of B with respect to its interpretations. For convenience, we repeat the postulates of $\mathsf{B}(L, \rho)$:

$\Rightarrow$ The first-order complete rules $\mathcal{T}_0$.

A: $$\frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, \Delta}{\Gamma \Vdash_{\rho(i)} \alpha}$$

$\Vdash_{\rho(i)}$ A closed derivation operator for each agent S_i.

To prove soundness, we first prove a preliminary lemma about the relation between deduction structure belief set and sequents of B.

LEMMA 4.2. *Let $\rho(i)$ be a set of deduction rules for agent S_i. If $\Gamma \Vdash_{\rho_i} \alpha$, then $[S_i]\Gamma \Rightarrow [S_i]\alpha$ is true in every $\mathsf{B}(L, \rho)$-model.*

Proof. Suppose to the contrary that $[S_i]\Gamma \Rightarrow [S_i]\alpha$ isn't valid; then by definition there must be some deduction structure $d_i = \langle B, \rho_i \rangle$ such that α isn't a member of $\text{bel}(d_i)$, and $\Gamma \subseteq \text{bel}(d_i)$. Since $\Gamma \Vdash_{\rho_i} \alpha$, α must be included in $\text{bel}(d_i)$ by the closure property of deduction structures, a contradiction.

The import of Lemma 4.2 is that it relates the notion of theoremhood for the indexed derivation operator to the notion of validity for the belief operator; in formal symbols, $\Gamma \Vdash_{\rho(i)} \alpha \rightarrow \mathsf{B}(L,\rho) \models [S_i]\Gamma \Rightarrow [S_i]\alpha$. It is the key step in proving the soundness of B.

THEOREM 4.3. (Soundness of B)

$$\vdash_{\mathsf{B}(L,\rho)} \Gamma \Rightarrow \Delta \quad \rightarrow \quad \mathsf{B}(L,\rho) \models \Gamma \Rightarrow \Delta$$

Proof. Consider a closed tableau for some theorem of B. We would like to show that the root of the tree is valid. We do this by showing that whenever a set of daughters is valid, the parent is valid, *i.e.*, the deduction rules preserve validity. Then if the axioms are valid, the root node (and indeed every node in the tableau) must be valid.

The first-order complete rules are valid, since every B-interpretation is a first-order valuation. By Lemma 4.2, the top sequent of the attachment rule is also valid. Hence every sequent in a tableau is valid.∎

Remarks. We have proven the soundness of B relative to a closed derivation operator for each agent. Note that the condition of soundness on an agent's rules was not used in the proof of this theorem.

To prove completeness, we first prove a lemma that is the converse of Lemma 4.2.

LEMMA 4.4. (Attachment) *If the (perhaps denumerably infinite) set* $\{[S_i]\Gamma', \neg[S_i]\Delta'\}$ *is* $\mathsf{B}(L,\rho)$*-unsatisfiable, then for some* $\delta \in \Delta'$ *and finite* $\Gamma \subseteq \Gamma'$, $\Gamma \Vdash_{\rho(i)} \delta$.

Proof. Assume that for all sentences $\delta \in \Delta'$ and all finite subsets $\Gamma \subseteq \Gamma'$, $\Gamma \nVdash_{\rho(i)} \delta$. Then we can construct a deduction structure $d_i =_{\mathrm{df}} \langle \Gamma, \rho(i) \rangle$, which has the property that no member of Δ' is in bel(d_i). Hence, for the $\mathsf{B}(L,\rho)$-model $m =_{\mathrm{df}} \langle v_0, \varphi, \mathsf{U}, \{\ldots d_i \ldots\}\rangle$ we have $m \models [S_i]\gamma$ for each $\gamma \in \Gamma$, and $m \not\models [S_i]\delta$ for each $\delta \in \Delta'$.∎

COROLLARY 4.5. *For some $\delta \in \Delta$,*

$$\mathsf{B}(L,\rho) \models [S_i]\Gamma \Rightarrow [S_i]\Delta \quad \rightarrow \quad \Gamma \not\vdash_{\rho(i)} \delta \quad .$$

THEOREM 4.6. (Completeness of B) *Let Γ and Δ be finite subsets of Γ' and Δ', respectively. Then*

$$\mathsf{B}(L,\rho) \models \Gamma' \Rightarrow \Delta' \quad \rightarrow \quad \vdash_{\mathsf{B}(L,\rho)} \Gamma \Rightarrow \Delta$$

Proof. Since the semantics of B is similar to that of a first-order language in which the modal atoms act like unanalyzable atomic sentences, the proof is a slight modification of the method in Smullyan [63] for the first-order system $\mathcal{T}_0$. We will show that if there is no closed tableau for any sequent $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite subsets of Γ' and Δ', then there is a tableau with an infinite branch all of whose sequents are B-satisfiable.

Suppose there is no closed tableau for any such sequent. Then we can construct an open tableau with an infinite branch b, using all the sentences Γ' and Δ'. Let $\Sigma_j \Rightarrow \Pi_j$ be the jth sequent of b; define the set W by $\cup_j\{\Sigma_j, \neg\Pi_j\}$. If we have constructed the open tableau in a systematic manner (see Smullyan [63], pp. 58–60, for one method), W will be a Hintikka set. By Theorem 4.1, we know that W is first-order satisfiable. We must show that it is also B-satisfiable.

W contains a (perhaps infinite) set of modal atoms $[S_i]\Gamma_i$ and negations of modal atoms $\neg[S_i]\Delta_i$ for each agent S_i. Since there is no closed tableau, there is no finite subset $\Gamma'_i \subseteq \Gamma_i$ such that $\Gamma'_i \not\vdash_{\rho(i)} \delta_i$ for any $\delta_i \in \Delta_i$. Hence, by the contrapositive of the attachment lemma, the set $\{[S_i]\Gamma_i, \neg[S_i]\Delta_i\}$ is satisfiable for any S_i.∎

The compactness of B follows immediately from the completeness theorem.

COROLLARY 4.7. (Compactness of B) *If a set of sentences of L^{B} is unsatisfiable, it has a finite unsatisfiable subset.*

5. Representational Extensions to B

This chapter covers a number of extensions to deductive belief logics that are useful for AI problem-solving systems: knowledge (as opposed to belief), a theory of ignorance, a representation for common belief, and a simple theory of situations. With each extension, we give a semantics in terms of models of B; the given axiomatizations are sound and complete with respect to these models, although we will not prove this explicitly.

In terms of the rest of the thesis, the theories of ignorance, common beliefs, and situations are important for understanding the solution to the Not-So-Wise Man Problem in Section 6.3, and the example in Appendix A. We collect the relevant tableau rules into a system B^+ that is summarized at the end of this chapter.

5.1 Knowledge

Often we will want to say of an agent's beliefs that they are *true*, that they correctly correspond to the current state of affairs. If the agents' language L is the same as L^{B}, we can do this for any particular belief p by simply stating $[S_i]p \supset p$: whenever S_i believes p, it is true. If every belief of an agent is true, then this becomes an axiom schema, valid for every sentence p. In tableau terms, we write the following rule.

$$K_0 : \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow \Delta}{\Sigma, [S_i]\Gamma, \Gamma \Rightarrow \Delta}$$

K_0 states that if all of Γ are believed, then they are also true. Using K_0, we can prove the axiom schema $[S_i]p \supset p$:

$$\begin{array}{l c} & \Rightarrow [S_i]p \supset p \\ \cline{2-2} I_1 & [S_i]p \Rightarrow p \\ \cline{2-2} K_0 & [S_i]p, p \Rightarrow p \\ & \times \end{array}$$

If we wish to consider the more general case in which the agents' language is not the same as L^{B}, then we must introduce a modal operator $\mathbf{T}$ into L^{B}, such that the intended meaning of $\mathbf{T}p$ is that p is true. Because p does not refer to a sentence of L^{B}, the normal Tarskian definition of truth can be axiomatized using $\mathbf{T}$ (see Moore [51], pp. 78–80).

Models for systems with the rule K_0 are the same as B-models, with the restriction that if $p \in \text{bel}(d_i)$, then $V(p) = \mathbf{t}$.

In the AI literature, true belief is often called knowledge; or to put it more exactly, a concept called *knowledge* is often defined, which has the same axiomatization as belief, except that the axiom schema $[S_i]p \supset p$ is assumed (see Hintikka [21], McCarthy [44], and Moore [51]). A competing convention in AI is to call sets of statements about the world "knowledge bases," even when the statements could be false (as in Levesque [38]). For the most part, we will use the former convention in this thesis, although in a few informal expositions where the word "belief" sounds awkward, we have substituted "knowledge." Any belief logic in this thesis can be converted into a corresponding knowledge logic by the addition of the rule K_0.

In epistemology, the branch of philosophy dealing with the study of knowledge, it is generally agreed that knowledge and true belief are not the same concept: for example, a distinction is made between beliefs that just happen to be true, and beliefs that are true because an agent has justifications for them. Issues of this sort will become important as attempts are made in AI to build more complicated cognitive structures (*e.g.*, that contain justifications for beliefs), and the concepts of knowledge and true belief will have to diverge. However, for many purposes it is reasonable to approximate knowledge as true belief.

5.2 Circumscriptive Ignorance

In many situations, representing what *isn't* the case often turns out to be a harder problem than representing what is. McCarthy (McCarthy and Hayes [45]) was the first to point this out as an important problem for AI; he called it the *qualification problem.* To illustrate it we use the example of the Missionaries and Cannibals puzzle (McCarthy [47]). A formalization of this puzzle usually consists of a set of sentences in a formal language that express the basic facts of the situation, such as the presence of a boat, three cannibals, and three missionaries. However, a formal statement of the puzzle is open to the following line of attack: someone who is looking for an easy way to solve it might say "well, just use the bridge upstream." Certainly, there is nothing in the formalization that says there isn't such a bridge; so we write down an axiom to exclude that. But then there are many other ways, an infinite number in fact, to get around the hidden assumption that the missionaries and cannibals are restricted to using just those objects presented in the puzzle, and further that the objects perform as expected, *i.e.*, the boat doesn't leak, and so on. How does one formalize these hidden assumptions? McCarthy's solution was to introduce a *circumscription schema.* The idea is that, once we have a first-order formalization that describes a situation, we can find an instantiation of the circumscription schema that will effectively state the assumptions that we automatically take to be part of the rules governing the situation.

A similar problem occurs with the representation of belief subsystems. In puzzles that involve reasoning about the beliefs of agents, there are often unstated conditions on the initial information given an agent, as well as on the information he can acquire. In the Wise Man Puzzle (see Section 6.3 for a full statement of this puzzle), it is common knowledge that each man can see his neighbors' spots but not his own. It is an unstated condition of the puzzle that this is the *only* knowledge that the wise men have about the initial situation. In effect, the knowledge that is available to the agents in the puzzle is being circumscribed; informally one would say "The only knowledge that an agent S has about proposition p are the facts F." If from F it is not possible for S to infer p (or $\neg p$), then S does not know whether p is true. In an adequate formalization of the Wise Man Puzzle, it must be possible to state conditions of this sort, and prove from them that each wise man is ignorant of the color of his own spot in the initial situation.

Let us give the name *circumscriptive ignorance* to those situations in which agents have only a limited amount of information available in deciding a particular question. Note that the language L^{B} is not expressive enough to represent circumscriptive ignorance. Suppose, for example, that we want to say that Sue believes our two favorite sentences P and Q, and all the consequences she can derive from them, *but no others*. We could start to list all of the things she doesn't believe: $\neg[Sue]\neg Q$, $\neg[Sue]\neg P$, and so on. Obviously, if there are an infinite set of primitive sentences, there is no finite way to complete such a list. To put this another way: in B it is possible to say that some sentences are in a belief set, and some aren't; but there is no finite way to say that a set of sentences are the *only* ones that are available in a belief set for deriving a particular belief.

To formalize circumscriptive ignorance, we add a *circumscription operator*[7] to L^{B}. It is written as $\langle S_i : \Gamma\rangle$, where Γ is a finite set of sentences of L^{B}, and used in expressions of the form $\langle S_i : \Gamma\rangle p$, where p is a sentence of L^{B}. The intended meaning is that p is derivable from Γ in the belief subsystem of agent S_i, that is, $\Gamma \Vdash_{\rho(i)} p$. Thus, the circumscription operator elevates the belief derivation process

[7] This operator is a variant of the one introduced in Konolige [31].

to a first-class entity of the language (as opposed to belief operators $[S_i]$, which simply state that certain sentences are or are not in the belief set).

While it may not be apparent at first glance, the circumscription operator is a powerful tool for representing situations of delimited knowledge. For example, to formally state the condition, "the only knowledge that agent S has about proposition p are the facts F," we could use

$$\langle S : F \rangle p \equiv [S]p \quad . \tag{5.1}$$

This assertion states that S believing p is equivalent to S being able to derive p from F. The forward implication is trivial, since it just says that p is derivable from F by agent S, *i.e.*, $[S]F \supset [S]p$. The reverse implication is more interesting, since it states p cannot be a belief of S *unless* it is derivable from F. This limits the information S has available to derive p to the sentences F, and thus gives the circumscriptive content of (5.1). Note that there is no way to formulate the reverse implication as a sentence of L^{B} using only belief operators.

An example of the use of the circumscription operator to represent circumscriptive ignorance is given in the solution to the Wise Man Puzzle in Section 6.3; readers who are uncertain about the consequences of an axiom such as (5.1) might wish to consult it. In the remainder of this section we will give an axiomatization of the circumscription operator, and discuss some limitations and possible extensions.

5.2.1 Axiomatization of the Circumscription Operator

For circumscriptive ignorance, we add the following formation rule to Definition 3.6 of L^{B}:

3. If p is a sentence and Γ is a finite set of sentences, then $\langle S_i : \Gamma \rangle p$ is a sentence.

Sentences of the form $\langle S_i : \Gamma\rangle p$ are called *circumscriptive atoms.* Note that, by this inductive definition, sentences that contain the circumscription operator can appear inside circumscriptive atoms. This allows agents to have beliefs about the circumscription of other agents' beliefs. However, we do not allow quantifying into the context of circumscriptive atoms—both Γ and p must be *sentences.*

We now give tableau rules for circumscriptive atoms.

$$Circ_1: \qquad \frac{\Sigma \Rightarrow \langle S_i : \Gamma\rangle p, \Delta}{\Gamma \Vdash_{\rho(i)} p}$$

$$Circ_2: \qquad \frac{\Sigma, \langle S_i : \Gamma\rangle p \Rightarrow \Delta}{\Gamma \not\Vdash_{\rho(i)} p}$$

Remarks. These rules are a straightforward formalization of the intended meaning of the operator in terms of belief deduction. The first says that the circumscriptive atom $\langle S_i : \Gamma\rangle p$ is true if $\Gamma \Vdash_{\rho(i)} p$, while the second says that it is false if $\Gamma \not\Vdash_{\rho(i)} p$,

Example. Suppose the agent Sue believes only the sentences P and $P \supset Q$ in a situation; we want to show that she doesn't believe R. Thus we want to prove the sequent $\langle Sue : P, P \supset Q\rangle R \equiv [Sue]R \Rightarrow \neg[Sue]R$.

$$\begin{array}{l}
C_1 \\ I_2
\end{array}
\frac{\dfrac{\langle Sue : P, P \supset Q\rangle R \equiv [Sue]R \Rightarrow \neg[Sue]R}{\langle Sue : P, P \supset Q\rangle R \supset [Sue]R, [Sue]R \supset \langle Sue : P, P \supset Q\rangle R \Rightarrow \neg[Sue]R}}{Circ_2\ \dfrac{\langle Sue : P, P \supset Q\rangle R \Rightarrow \neg[Sue]R}{P, P \supset Q \not\Vdash_{\rho(i)} R} \qquad N_2\ \dfrac{\Rightarrow [Sue]R, \neg[Sue]R}{\begin{array}{c}[Sue]R \Rightarrow [Sue]R \\ \times\end{array}}}$$

$P, P \supset Q \not\Vdash_{\rho(i)} R$ is valid if the rules $\rho(i)$ are sound, and so both branches of the tableau close. Note that only the reverse implication half of the equivalence was needed.

It is an interesting exercise to show that $\langle S_i : \Gamma\rangle p \Rightarrow [S_i]\Gamma \supset [S_i]p$ holds, but the converse doesn't.

THEOREM 5.1. $\vdash_{\mathsf{B}+Circ_1+Circ_2} \langle S_i : \Gamma\rangle p \Rightarrow [S_i]\Gamma \supset [S_i]p$

Proof. We have the following two tableaux for this sentence.

$$\begin{array}{rc} I_1 & \langle S_i : \Gamma\rangle p \Rightarrow [S_i]\Gamma \supset [S_i]p \\ \hline & \langle S_i : \Gamma\rangle p, [S_i]\Gamma \Rightarrow [S_i]p \\ A & \hline \\ & \Gamma \Vdash_{\rho(i)} p \end{array}$$

$$\begin{array}{rc} I_1 & \langle S_i : \Gamma\rangle p \Rightarrow [S_i]\Gamma \supset [S_i]p \\ \hline & \langle S_i : \Gamma\rangle p, [S_i]\Gamma \Rightarrow [S_i]p \\ Circ_2 & \hline \\ & \Gamma \nVdash_{\rho(i)} p \end{array}$$

Either p is derivable from Γ using the rules $\rho(i)$, or it isn't. In either case, one of these tableaux closes.▮

THEOREM 5.2. $\nvdash_{\mathsf{B}(L_0,\rho)+Circ_1+Circ_2} [S_i]\Gamma \supset [S_i]p \Rightarrow \langle S_i : \Gamma\rangle p$

Proof. We have the following tableau:

$$I_2 \quad \dfrac{[S_i]\Gamma \supset [S_i]p \Rightarrow \langle S_i : \Gamma\rangle p}{Circ_1\ \dfrac{[S_i]p \Rightarrow \langle S_i : \Gamma\rangle p}{\Gamma \Vdash_{\rho(i)} p} \qquad Circ_1\ \dfrac{\Rightarrow [S_i]\Gamma, \langle S_i : \Gamma\rangle p}{\Gamma \Vdash_{\rho(i)} p}}$$

For some choices of the rules $\rho(i)$ and sentences Γ and p, the belief derivation will not hold, and so this tableau will not close.▮

5.2.2 *Limitations and Extensions*

A circumscription atom $\langle S_i : \Gamma\rangle p$ has an argument p, so that it is possible to assert that Γ is the only information available for deriving p. The ability to name a sentence p that we are interested in is often very useful, since it lets us apply circumscriptive ignorance to a particular belief. In effect, we can disregard all the other information an agent may have about the world, and say that, with respect to the particular sentence p, Γ are all the beliefs that need be considered. In this way, the circumscription operator solves what Moore [51] has called the *compartmentalization problem*: the problem of representing the ability of agents to exclude from consideration all but a small subset of their beliefs when trying to answer a belief query.

However, it may be the case that we want to circumscribe the *total* set of beliefs of an agent, that is, to say that the base set of an agent must be contained in a finite set Γ. This cannot be done in a finite manner with the circumscription operator as defined, since we are forced to give a particular argument to the operator. A variation of the circumscription operator that does not take an argument is useful here. Suppose we let the sentence $\langle S_i : \Gamma \rangle$ mean: "the *most* that S_i could have in his base set are the sentences Γ." The following rule axiomatizes this circumscription operator.

$$Circ_1' : \qquad \frac{\Sigma, [S_i]p, \langle S_i : \Gamma \rangle \Rightarrow \Delta}{\Gamma \not\Vdash_{\rho(i)} [S_i]p}$$

$Circ_2$ states that if p does not follow from Γ according to agent S_i, one of $[S_i]p$ or $\langle S_i : \Gamma \rangle$ is false: either p is not one of S_i's beliefs, or there are other sentences besides Γ in the base set. $Circ_1'$ can be used to prove that an agent doesn't believe something; for example, we could show that if S_i has at most the sentence P in his base set, he doesn't believe Q:

$$\begin{array}{ll} N_1 & \dfrac{\langle S_i : P \rangle \Rightarrow \neg[S_i]Q}{\langle S_i : P \rangle, [S_i]Q \Rightarrow} \\ Circ_1' & \dfrac{}{P \not\Vdash_{\rho(i)} Q} \end{array}$$

Another extension we might wish to make is to allow quantifying into the argument of the circumscription operator, as we do for belief operators in Chapter 9. We would then be able to formalize situations like the following: "The only information Sue has about her local chapter of the DAR is that there is someone whom she believes to be a member." However, the axiomatization of quantifying-in for the circumscription operator turns out to be complex, and we have not been able to find any useful proof methods for it.

There is, however, an alternative to quantifying-in that is more in the spirit of our original work on circumscriptive ignorance (in Konolige [31]). That is, we assume that the sentences Γ and p of a circumscriptive atom are a *description* of the beliefs of an agent, rather than referring directly to the beliefs themselves. So, for example, we would rephrase (5.1) as

$$\langle S : [S]F\rangle[S]p \equiv [S]p \quad . \tag{5.2}$$

Note that the agent argument in the circumscription operator is now redundant and can be dropped. With this new operator we have a way of representing quantified-in descriptions of an agent's beliefs for the purposes of circumscriptive ignorance. For example, we might state

$$\langle \exists x.[S]Px\rangle[S]Pa \equiv [S]Pa \quad , \tag{5.3}$$

which says that the only belief S has about Pa is a belief Px for some individual x. The penalty we pay for this increased expressivity is a more difficult axiomatization. While the tableau rules $Circ_1$ and $Circ_2$ related circumscriptive atoms to belief derivation in an agent's subsystem, the modified notion of circumscription must be axiomatized in terms of the proof process of B itself. We would use the following tableau rules.

$$Circ_1'': \qquad \frac{\Sigma \Rightarrow \langle\Gamma\rangle p, \Delta}{\Gamma \vdash_{\mathsf{B}+Circ_1''+Circ_2''} p}$$

$$Circ_2'': \qquad \frac{\Sigma, \langle\Gamma\rangle p \Rightarrow \Delta}{\Gamma \nvdash_{\mathsf{B}+Circ_1''+Circ_2''} p}$$

$Circ''_1$ and $Circ''_2$ involve the logical deduction operator on the system $\mathsf{B} + Circ''_1 + Circ''_2$, instead of belief deduction in an agent's subsystem. These rules can only be implemented as deduction rules if the theory of $\mathsf{B} + Circ''_1 + Circ''_2$ is decidable.

5.3 Common Beliefs

A common belief is a belief that every agent has, and that every agent believes every agent has, and so on to arbitrary levels of belief nesting. Common beliefs occur frequently in the statement of puzzles such as the Wise Man Puzzle, and are also essential in the understanding of communication among agents (see, for example, Clark [6]).

Common beliefs have a natural interpretation in terms of the deduction model. Let us use the notation $[S_0]p$ to say that p is a common belief. One condition we want to impose on the truth of $[S_0]p$ is that p be contained in the belief set of every agent S_i. A further condition is that every agent must believe that p is a common belief; and so $[S_0]p$ must also be present in each agent's belief set. Taken together, these two conditions allow common beliefs to percolate down to arbitrary levels of belief nesting, for if an agent believes $[S_0]p$, he will believe every agent believes $[S_0]p$ (by the second condition), and so on. Note that modal operators containing S_0 have exactly the same form as those using "real" agent names S_i, $i \geq 1$. We are thus led to consider S_0 as a fictional agent whose beliefs are common beliefs.

McCarthy [44] was the first to propose that common knowledge be represented by the use of a fictitious agent FOOL whose knowledge "any fool" would know. He uses a possible-world semantics for knowledge, and so all consequences of common knowledge are also known. The theory of common belief presented here uses an obviously similar approach; it differs only in that common belief rather than common knowledge is axiomatized (common beliefs need not be true), and in having a deduction structure semantics, so that common beliefs need not be closed under logical consequence.

The concept of common beliefs presented here can be generalized to mutual belief, in which a subset of the agents share beliefs (see Appelt [1], pp. 51–54).

The semantics of common belief in B-models is straightforward; we simply modify the valuation function V for belief atoms to reflect the fact that every agent has the common belief. To Definition 4.5 we add the following rule:

3. $V([S_0]p) = \mathrm{t}$ *iff* for all $i \geq 1$, $p \in \mathrm{bel}(d_i)$ and $[S_0]p \in \mathrm{bel}(d_i)$.

5.3.1 Axiomatization of Common Beliefs

In the language L^{B}, we allow $[S_0]$ to be a belief operator. An atom of the form $[S_0]p$ is called a *common belief atom.*

The tableau rule for common belief is a modification of the attachment rule. We also place a condition on the common belief derivation rules $\rho(0)$.

A^{CB} : $\dfrac{\Sigma, [S_0]\Lambda, [S_i]\Gamma \Rightarrow [S_i]\alpha, \Delta}{[S_0]\Lambda, \Lambda, \Gamma \Vdash_{\rho(i)} \alpha}$

$\Vdash_{\rho(0)}$ A closed derivation process for S_0, such that $\rho(0) \subseteq \rho(i)$ for every i.

Remarks. The attachment rule A^{CB} differs from A in that common beliefs are carried down to be used in belief derivation. Note that both Λ and $[S_0]\Lambda$ are present, fulfilling the two conditions that we want common beliefs to satisfy. The common belief derivation rules $\rho(0)$ are those that any agent must know. The analogy to McCarthy's "any fool" concept is very strong here, because we have even endowed this fictitious agent with his own rule set.

We prove one theorem about $\mathsf{B} + A^{\mathrm{CB}}$: if p is a common belief, then it is a common belief that this is so.

THEOREM 5.3.

$$\vdash_{\mathsf{B}+A^{\mathrm{CB}}} [S_0]p \Rightarrow [S_0][S_0]p$$

Proof.

$$A^{\mathrm{CB}} \quad \dfrac{[S_0]p \Rightarrow [S_0][S_0]p}{\dfrac{[S_0]p, p \Vdash_{\rho(0)} [S_0]p}{\times}}$$

Some examples of the use of common beliefs are given in the solution to to Wise Man Puzzle and the Not-so-Wise Man Problem (Section 6.3).

5.4 A Simple Theory of Situations

We now describe and axiomatize a simple theory of situations. This theory will be used in the solution of the Wise Man Puzzle, and illustrates one method of formalizing the acquisition of new beliefs through time. The theory actually fits the concept of knowledge better than belief, since we assume that beliefs, once held, are never relinquished; we do not deal with belief revision in this simple theory.

The basic approach is to define an infinite set of situations $t_1, t_2, \ldots$ that are fully ordered with respect to time, so that $t_1 < t_2 < \ldots$. Each situation describes a state of the world, including agents' beliefs; that is, it is a B-model. We assume that if S believes p in situation t_i, he also believes p in every situation t_j such that $j \geq i$. Another assumption is that an agent's belief derivation rules do not change over time: if he uses the rules $\rho(i)$ in situation t_1, then these are the rules he uses in every situation.

5.4.1 Axiomatization of Situation Operators

To formalize this theory, we add a sequence of situations $t_1, t_2, \ldots$ to L^{B}, and include them as an argument to the belief operator. $[S, t]p$ means that agent S believes p in the situation t. The tableau rule for these belief operators is the following modification of the attachment rule.

$$A^{\mathrm{S}}: \quad \frac{\Sigma, [S_i, t_{k_1}]\Gamma_1, [S_i, t_{k_2}]\Gamma_2, \ldots \Rightarrow [S_i, t_n]\alpha, \Delta}{\Gamma_1, \Gamma_2, \ldots \Vdash_{\rho(i)} \alpha}, \quad \text{where } t_{k_j} \leq t_n.$$

Remarks. The attachment rule A^{S} allows an agent's beliefs in situations previous to a situation t_n to contribute to the derivation of beliefs in t_n; hence beliefs are cumulative with time, as we now prove.

THEOREM 5.4. *If $t_k \leq t_n$, then*

$$\vdash_{\mathrm{B}+A^{\mathrm{S}}} [S_i, t_k]p \Rightarrow [S_i, t_n]p \quad .$$

Proof.

$$A^{\mathrm{S}} \quad \frac{[S_i, t_k]p \Rightarrow [S_i, t_n]p}{p \Vdash_{\rho(i)} p}$$

This tableau closes by the definition of $\Vdash$. ∎

Although we have chosen to make ρ a function of the agent only, and not the situation, it is a simple extension of this theory to allow for agents having different derivation rules in different situations. The attachment rule A^{S} remains the same, except that the index on ρ becomes $\rho(i, n)$ for situation t_n.

The theory of situations given here was first presented by McCarthy [44] in solving the Wise Man Puzzle, who formalized it by axiomatizing its possible-world semantics. Sato [61] developed a Gentzen system axiomatization of both common knowledge and McCarthy's situational theory, using a propositional modal language.

5.4.2 Semantics

The models of situations are defined in terms of B-models. To each situation t_k we assign a B-model m_k. Because beliefs are cumulative in time, there is a restriction on allowable m_k's: the belief set of an agent in situation t_j must include his belief set in situation t_i, if $i \leq j$. We call the resultant models B^+-models.

The valuation of belief atoms in B^+-models is given by the following rules:

2. If $S_i \neq S_0$, $V([S_i, t_k]p, m) = \mathbf{t}$ *iff* $p \in \mathrm{bel}(d_i^k)$, where $d_i^k \in D^k$.
3. $V([S_0, t_k]p) = \mathbf{t}$ *iff* for all $i \geq 1$, $p \in \mathrm{bel}(d_i^k)$ and $[S_0, t_k]p \in \mathrm{bel}(d_i^k)$.

5.5 The System B^+

We collect the theories of common belief, circumscriptive ignorance, and simple situations into a single deductive logic family B^+. Its language L^{B^+} includes a sequence of situation $t_1, t_2, \ldots$, circumscriptive atoms $\langle S_i : \Gamma\rangle p$, situational belief atoms $[S_i, t_j]p$, and common belief atoms $[S_0, t_j]p$.

Let Z be a set of positive belief atoms; we define the set of sentences $Y(S_i, t_j, Z)$ as the least set satisfying:

1. If $[S_i, t_k]p \in Z$, where $t_k \leq t_j$, then p is in $Y(S_i, t_j, Z)$.
2. If $[S_0, t_k]p \in Z$, where $t_k \leq t_j$, then p and $[S_0]p$ are in $Y(S_i, t_j, Z)$.

$Y(S_i, t_j, Z)$ can be thought of as all those sentences that the literal set Z asserts to be in the belief set of agent S_i at time t_j.

We now define the system B^+.

DEFINITION 5.1. *The system* $\mathsf{B}^+(L_0, \rho)$ *has the following postulates.*

$\Rightarrow$ *The first-order complete rules* $\mathcal{T}_0$.

$$A^+: \quad \frac{\Sigma, Z \Rightarrow [S_i, t_n]\alpha, \Delta}{Y(S_i, t_n, Z) \Vdash_{\rho(i)} \alpha}$$

$$Circ_1: \quad \frac{\Sigma \Rightarrow \langle S_i, t_k : \Gamma\rangle p, \Delta}{\Gamma \Vdash_{\rho(i)} p}$$

$$Circ_2: \quad \frac{\Sigma, \langle S_i, t_k : \Gamma\rangle p \Rightarrow \Delta}{\Gamma \nVdash_{\rho(i)} p}$$

$\Vdash_{\rho(i)}$ *A closed derivation process for each agent* S_i*, such that* $\rho(0) \subseteq \rho(i)$ *for every* i.

The nonintrospective version of B^+, called BK^+, substitutes indexed sequents for belief derivation in the rule A^+:

$$A_K^+: \quad \frac{\Sigma, Z \Rightarrow [S_i, t_n]\alpha, \Delta}{Y(S_i, t_n, Z) \Rightarrow_i \alpha}$$

6. Three Problems in the Representation of Belief

In this chapter we use the logic B to solve some problems in the representation of belief. These problems illustrate in some detail the types of incompleteness an agent may be subject to in reasoning about his beliefs: *resource incompleteness*, *fundamental logical incompleteness*, and *relevance incompleteness*. In formulating answers to the problems, we have tried to steer clear of solutions that are trivial, in the sense that they solve the representation problem, but only at the expense of excluding types of reasoning that might be expected to occur. For example, in the chess problem, it would be a sufficient but unrealistic solution to credit each player with no deduction rules at all. Instead, we try to find rules that allow a resource-limited amount of reasoning about the game to take place.

6.1 The Chess Problem

Suppose an agent knows the rules of chess. It does not necessarily follow that he knows whether White has a winning strategy or not.

The chess problem, on the face of it, seems hardly to be a representational problem at all. Certainly its statement is *true*: no agent, human or otherwise, can possibly follow out all the myriad lines of chess play allowed by the rules to determine whether White has a strategy that will always win. What kind of model of belief would lead us to expect an agent to know whether White has a winning strategy? The answer is, of course, a model that does not take resource limitations into account in representing an agent's reasoning about the consequences of his beliefs. Within such a model, we could establish the following line of argument.

> *Chess is a finite game,[8] and so it is possible, in theory, to construct a complete, finite game tree for chess, given the rules of the game. The question of White's having a winning strategy or not is a property of this finite game tree. If for every counter Black makes, White has a move that will lead to a win, then White has a winning strategy. Thus, White's having a winning strategy is a consequence of the rules of chess that can be derived in a finite number of simple steps. One simply constructs the finite game tree using the rules of chess, and then checks the tree for the existence of such a strategy. If an agent believes all the logical consequences of his beliefs, then an agent who knows the rules of chess will, by the reasoning just given, also know whether White has a winning strategy or not.*

The chess problem is thus a problem in representing reasoning about beliefs in the face of resource limitations. The inference steps themselves are almost trivial; it is a simple matter to show that a move is legal, and hence to construct positions that follow *via* a legal move from a given position. But, although the individual inferences are easy, the number of them required to figure out whether White has a forced win is astronomical and beyond the computational abilities of any agent. We call this behavior *resource-limited incompleteness.* A suitable model of belief must be able to represent situations of this sort, in which an agent possesses the inferential capability to derive some consequence of his beliefs, but simply does not have the resources to do so.

[8] The finiteness of chess is assured by the rule that, if 50 moves occur without a pawn advance or piece capture, the game is a draw.

6.1.1 Solution to the Chess Problem

To approach this problem, we need to represent the game in a first-order language. Because the ontology of chess involves rather complicated objects (pieces, board positions, moves, histories of moves) we will not give a complete formalization, but rather sketch in outline how this might be done in a tableau system. The basic idea is to let the structure of the tableau mimic the structure of the game tree. At each node in the tableau, there is an expression that indicates the depth of the game tree search. By proscribing deductions below a fixed level of the tree, it is possible to do a limited amount of reasoning about a chess position.

We use a multisorted first-order language L_c for the base language L_0. The key sorts will be those for players (S_w or S_b), moves, and boards. The particular structure of the sort terms is not important for the solution of this problem, but they should have the following information. A board contains the position of all pieces, and a history of the moves that were made to get to that position. This is important because we want to be able to find all legal moves from a given position; to do this, we have to have the sequence of moves leading up to the position, since legal moves can be defined only in terms of this sequence. For example, castling can only occur once, even if a player returns to the position before the castle; more importantly, there are no legal moves if 50 moves have been made without a capture or pawn advancement (this is what makes chess a finite game). A move contains enough information so that it is possible to compute all successor boards, that is, those resulting from legal moves.

The game tree is a useful concept in exploring game-playing strategies. This is a finite tree (for finite games like chess) whose nodes are board positions, whose arcs are labeled by legal moves, and whose branches are all possible complete games. A terminal node of the tree ends in either a win for White or Black, or a draw. The *game-theoretic value* of a node for a player is either 1 (a win), 0 (a draw), or -1 (a loss), based on whether that player can force a win or a draw, or his opponent can force a win. We use the predicate $M(p,b,k,l,r)$ to mean that board b has value k for player p. The argument l is a depth-of-search indicator, and shows the maximum depth of the game tree that the value is based on. We include the argument r so that M can represent heuristic information about the value of a node; when $r = \mathbf{f}$, k is the player's subjective estimate of the value of the node, *i.e.*, he has not searched to all terminal nodes of the game tree. If $r = \mathbf{t}$, then k is the game-theoretic value of the board.

We take the formal interpretation of boards, players, and the M predicate to be the game of chess, so that L_c is a partially interpreted language. The rules of the game of chess strictly specify what the game tree and its associated values will be; hence, each predication $M(p,b,k,l,\mathbf{t})$ or its negation is a valid consequence of these interpretations. Any agent who knows the rules of chess, and who has the concept of game trees, will know the game-theoretic value of every node if his beliefs are consequentially closed. In particular, he will believe either $M(S_w, I, 1, l, \mathbf{t})$ or $\neg M(S_w, I, 1, l, \mathbf{t})$, where I is the initial board; and so he will know whether White has an initial forced win or not.

We represent agents' knowledge of chess by giving tableau rules for L_c. The rules $\mathcal{T}_c$ presented below are one possible choice.

$$Ch_1: \quad \frac{\Gamma \Rightarrow M(p,b,k,l,r), \Delta}{\Gamma \Rightarrow M(p,b_1,k_1,l_1,r_1), \Delta \quad \Gamma \Rightarrow M(p,b_2,k_2,l_2,r_2), \Delta \quad \cdots \quad \Gamma \Rightarrow M(p,b_n,k_n,l_n,r_n), \Delta},$$

where b_1–b_n are *all* the legal successor boards to b
p's opponent is to move on b
k is the minimum of k_1–k_n
l is 1+ the maximum of l_1–l_n
r is $\mathbf{t}$ *iff* all of r_1–r_n are $\mathbf{t}$

$$Ch_2: \quad \frac{\Gamma \Rightarrow M(p,b,k,l,r), \Delta}{\Gamma \Rightarrow M(p,b_1,k_1,l_1,r_1), \Delta \quad \Gamma \Rightarrow M(p,b_2,k_2,l_2,r_2), \Delta \quad \cdots \quad \Gamma \Rightarrow M(p,b_n,k_n,l_n,r_n), \Delta},$$

where b_1–b_n are *all* the legal successor boards to b
p is to move on b
k is the maximum of k_1–k_n
l is 1+ the maximum of l_1–l_n
r is $\mathbf{t}$ *iff* all of r_1–r_n are $\mathbf{t}$

Ch_3 : $\Gamma \Rightarrow M(p, b, k, 0, \mathbf{t}), \Delta$, where $k = 1$ if p has a checkmate on his opponent on board b; $k = 0$ if board b is a draw; and $k = -1$ if p's opponent has a checkmate.

Ch_4 : $\Gamma \Rightarrow M(p, b, k, 0, \mathbf{f}), \Delta$, where k is any number between -1 and 1

Ch_1 axiomatizes nodes in the game tree where p's opponent moves. The value of such a node is the minimum of the values of its successor nodes. The argument l is the maximum depth of the subtree searched. r will be $\mathbf{t}$ only if all the subtrees have been searched to leaf nodes. Ch_2 is similar to Ch_1, except p moves, and the maximum of the successor values is chosen.

Ch_3 is the rule for terminal nodes of the tree. Ch_4 is a rule for heuristic evaluation of any node: note that the last argument to M is $\mathbf{f}$, which indicates that a terminal node has not been reached. Each agent may have his own particular heuristics for evaluating nonterminal nodes; we can accommodate this by changing the values for k in Ch_4.

As an example of the use of these rules, consider the following tableau proof.

$$Ch_1\ \dfrac{\Rightarrow M(S_w, b, 1, 2, \mathbf{t})}{\begin{array}{ccc} \Rightarrow M(S_w, b_1, 1, 0, \mathbf{t}) & & \Rightarrow M(S_w, b_5, 1, 0, \mathbf{t}) \\ \times & Ch_2\ \dfrac{\Rightarrow M(S_w, b_2, 1, 1, \mathbf{t})}{\begin{array}{cc} \Rightarrow M(S_w, b_3, 0, 0, \mathbf{t}) & \Rightarrow M(S_w, b_4, 1, 0, \mathbf{t}) \\ \times & \times \end{array}} & \times \end{array}} \tag{6.1}$$

This is a proof that the board b has a value 1 for White, searching to all terminal nodes. Boards b_1, b_2, and b_3 all have value 1, so an application of rule Ch_1 yields that value 1 for b (it is Black's turn to move on b). Boards b_1 and b_5 are terminal nodes that are checkmates for White. There are two legal moves from board b_2;

one ends in a draw (b_3), the other in a win for White (b_4). Since it is White's turn to move, rule Ch_2 applies.

The structure of this tableau proof mimics exactly the structure of the game tree from the board b. Indeed, for any subtree of the complete game tree of chess whose root is the board b with value k for player p, there is a corresponding proof of $M(p, b, k, l, \mathrm{t})$ using the rules $\mathcal{T}_c$. In particular, if one of $M(S_w, I, 1, l, \mathrm{t})$, $M(S_w, I, 0, l, \mathrm{t})$, or $M(S_w, I, -1, l, \mathrm{t})$ is true, there is a proof of this fact. Hence the rules $\mathcal{T}_c$ are sufficient for a player to reason whether White has a forced initial win or not, given an infinite resource bound for derivations. If we model agents as having the rules $\mathcal{T}_c$, so that $\mathcal{T}_c \subseteq \tau(\nu)$ for every view ν, each agent would know this.

A simple modification of the rules Ch_1 and Ch_2 can restrict exploration of the entire game tree, while still allowing agents to reason about game tree values using the heuristic axioms Ch_4, or the terminal node axioms Ch_3 if the game subtree is small. All that is necessary is to add the condition that no rule is applicable when the depth l is greater than some constant N. S_w would still be able to reason about the game to depths less than or equal to N, but he could go no further. In this way, a deductively closed system can represent a resource-limited derivation process. The revised rules are:

Ch'_1 : Ch_1, with the condition that $l \leq N$.
Ch'_2 : Ch_2, with the condition that $l \leq N$.

With these rules, the proof of (6.1) would still go through for $N \geq 2$, but a proof of $M(S_w, I, 1, l, \mathrm{t})$ could not be found if N were low enough to stop search at a reasonable level of the game tree.

The solution to the chess problem illustrates the ability of the deduction model to represent resource bounds by the imposition of constraints on deduction rules. There are other workable constraints for this problem besides depth cutoff:

for example, the number of nodes in the tree being searched could be kept below some minimum. Because the structure of proofs mimics the game tree, any cut-off condition that is based on the game tree could be represented by appropriate deduction rules.

6.2 The Syllogism Problem

Subjects are given the following two pairs of syllogistic premises:[9]

(1)a. *Some of the beekeepers are artists.*
b. *None of the chemists are beekeepers.*

(2)a. *Some of the artists are beekeepers.*
b. *None of the beekeepers are chemists.*

In the first case, most subjects failed to reach any valid conclusion; in the second, most subjects responded correctly with the conclusion:

(2′) *Some of the artists are not chemists.*

The two sets of premises are logically equivalent, although the form of each is different. In the second set of premises, the conclusion seems to follow almost automatically, perhaps because it is so similar to the familiar transitivity inference of equality: "if A is B, and B is C, then A is C." Aristotle called syllogisms with such obvious transitivity "perfect." Johnson-Laird and Steedman [25] dubbed the general dependence of the form of the conclusion on the form of the premises the *figural effect*; in the above example the "figure" of the first set of premises is $\frac{\text{B–A}}{\text{C–B}}$, that of the second set $\frac{\text{A–B}}{\text{B–C}}$. The figural-effect theory says that conclusions of the form C–A (*e.g.*, *none of the chemists are artists*) are more likely in the first case, while in the second A–C would be expected. Since there are no valid conclusions

[9] The syllogisms and experimental results are from Johnson-Laird and Steedman [25].

of the form C–A, the figural-effect theory correctly predicts that subjects will have a harder time coming to a valid conclusion when presented with the first set of premises.

Why is the syllogism problem a representation problem for belief, and how does it differ from the chess problem? Let us suppose that a subject is asked to believe that the premises of a syllogism are true. According to the symbol-processing theory of belief, this means that, for each of the premises, he forms an internal string of symbols that represents the meaning of the premise. It is not important what the exact nature of the representation is, but it is important that it accurately reflect the truth-conditions imposed by the premise, otherwise it would not be correct to say that the subject actually believed the premise. So let us assume that the subject has internal symbol strings that count as beliefs in the two premises of a syllogism. If the subject comes to different conclusions when asked to believe the premise pairs (1) and (2), then the symbol strings he uses as beliefs must be different in each case, even though they have the same truth-conditions. The subject's inferential process, operating on the two different symbol strings, produces different conclusions for each syllogism.

The important point to note in this symbol-processing reconstruction of the syllogism problem is that, when the inference process employed by the subject is *syntactic*—depending on the form of his beliefs rather than their truth-conditions—then it might be *incomplete* from a logical point of view. The incompleteness of syllogistic reasoning seems to be of a different sort from that of the chess problem, which was resource-limited. Syllogisms involve only two simple premises; it seems highly unlikely that the failure of subjects to conclude anything from the pair of premises (1) is a consequence of limited space and time in which to draw conclusions. Rather, this experiment suggests that the subject's inferential process, even when given adequate resources, is just not powerful enough in terms of its ability to arrive at simple logical conclusions. To apply an analogy from high-school algebra:

a student who is confronted with the equation $x + a = b$ and asked to solve for x won't be able to do so if he doesn't know the rule that subtracting equals from each side leaves the equation valid. It is not that the student lacks sufficient mental resources to solve this problem; rather, his rules of inference for dealing with equational theories are logically incomplete. To contrast this type of incompleteness with the resource-limited incompleteness described in the chess problem, we call it *fundamental logical incompleteness.*

The evidence from psychological experiments is only *suggestive* of the hypothesis that human belief inference is incomplete, because it involves the assumption that an agent has an internal representation that counts for a belief in the facts described by the English sentences of the syllogism. To the extent that we grant this assumption (certainly a controversial one in the psychological arena), the syllogism problem is evidence for the hypothesis that, if one wants to model the behavior of agents who possess beliefs but are imperfect reasoners, then it had better be the case that the model can account for an agent's not believing some "easy" logical consequences of his beliefs, with respect to which resource limitations are not a problem. This cannot be a characteristic of models of belief that, like the possible-world model, use closure under logical implication to model reasoning about beliefs.

6.2.1 Solution to the Syllogism Problem

To solve this problem, we need to find deduction rules that are sensitive, in the right way, to the form of expression of the premises of a syllogism. In addition, we want such a system of rules to be extensible to capture other types of commonsense reasoning based on quantified statements, *e.g.*, particularizing a universal statement so that it applies to a given individual. This criterion is to prevent trivial solutions in which the syllogistic evidence is accomodated, but at the expense of positing a very rigid and unextensible framework.

First, let us translate the syllogisms into a logical language. Consider a base language L_s that contains three monadic predicates:

Ax :	x is an artist.
Bx :	x is a beekeeeper.
Cx :	x is a chemist.

Using these predicates, the syllogisms can be phrased as sentences of L_s. There are actually many ways to capture the semantic content of the syllogisms in L_s, but we are trying to preserve something of their original syntax in the translation, given the imperfect match between English and first-order languages. One fairly straightforward translation is the following:

(1)a.	$\exists x.\ Bx \wedge Ax$	Some of the beekeepers are artists.
b.	$\forall x.\ Cx \supset \neg Bx$	None of the chemists are beekeepers.
(2)a.	$\exists x.\ Ax \wedge Bx$	Some of the artists are beekeepers.
b.	$\forall x.\ Bx \supset \neg Cx$	None of the beekeepers are chemists.
(2′)	$\exists x.\ Ax \wedge \neg Cx$	Some of the artists are not chemists.

We have been careful to reflect the order in which classes appear in the original syllogisms: note the difference between the premises of (1a) and (2a). Although the semantic content of the logical translation is trivially the same, the order of the conjuncts is different.

The reader may wish to verify that the conclusion actually does logically follow from each of the premise sets. Using the rules $\mathcal{T}_0$, here is a proof from the premises (1):

$$
\begin{array}{rc}
 & \exists x.\ Ax \wedge Bx, \forall x.\ Bx \supset \neg Cx \Rightarrow \exists x.\ Ax \wedge \neg Cx \\ \cline{2-2}
E_2 & Ae \wedge Be, \forall x.\ Bx \supset \neg Cx \Rightarrow \exists x.\ Ax \wedge \neg Cx \\ \cline{2-2}
U_1 & Ae \wedge Be, Be \supset \neg Ce \Rightarrow \exists x.\ Ax \wedge \neg Cx \\ \cline{2-2}
E_1 & Ae \wedge Be, Be \supset \neg Ce \Rightarrow Ae \wedge \neg Ce \\ \cline{2-2}
C_1 & Ae, Be, Be \supset \neg Ce \Rightarrow Ae \wedge \neg Ce \\ \cline{2-2}
I_2 & \begin{array}{ccc}
Ae, Be \Rightarrow Be, Ae \wedge \neg Ce & C_2 & \begin{array}{cc}
\multicolumn{2}{c}{Ae, Be, \neg Ce \Rightarrow Ae \wedge \neg Ce} \\ \hline
Ae, Be, \neg Ce \Rightarrow \neg Ce & Ae, Be, \neg Ce \Rightarrow Ae \\
\times & \times
\end{array} \\
\times & &
\end{array}
\end{array}
$$

The proof for premises (2) is similar, and is actually given as an example on page 45.

The rules $\mathcal{T}_0$ are thus too strong to model agents who fail to deduce all valid conclusions from the syllogism premises. They are, for example, insensitive to the order of conjuncts in quantified sentences like (1a) and (2a). In looking at the above tableau, the inappropriateness of $\mathcal{T}_0$ seems to be that its rules are too fine-grained. Rather than considering the two premise sentences as single entities, and drawing an immediate conclusion from them, $\mathcal{T}_0$ breaks each sentence into finer and finer components, until the atomic level is reached. At this point the character of the original sentence is lost. In terms of absolute theorem-proving power, this is an admirable property, because we don't want the accidental expression of a sentence to get in the way of its semantic content. However, in trying to model incomplete syllogistic deduction, it is too powerful a method.

Our strategy in solving this problem is to chunk together several of the simple steps of $\mathcal{T}_0$ into one deduction rule. By creating "macro" deduction rules of this sort to handle quantified statements, we can arrive at a system that is incomplete in precisely the way necessary to account for the syllogistic evidence of the figural effect. Recall that, in general form, the figural effect says that a conclusion of the form C–A is more likely to be drawn for a syllogism that has the form $\frac{\text{B-A}}{\text{C-B}}$, if such a conclusion is valid. The order that the premises are presented in is not important.

In place of the quantification rules U and E, consider the following tableaux schema:

$$V_1: \quad \frac{\Gamma,\ \exists x.\, \alpha(x) \land \beta(x),\ \forall y.\, \beta(y) \supset \gamma(y) \Rightarrow \Delta}{\Gamma,\ \exists x.\, \alpha(x) \land \beta(x),\ \forall y.\, \beta(y) \supset \gamma(y),\ \exists x.\, \alpha(x) \land \gamma(x) \Rightarrow \Delta}$$

$$V_2: \quad \frac{\Gamma,\ \forall x.\, \alpha(x) \supset \beta(x),\ \forall y.\, \beta(y) \supset \gamma(y) \Rightarrow \Delta}{\Gamma,\ \forall x.\, \alpha(x) \supset \beta(x),\ \forall y.\, \beta(y) \supset \gamma(y),\ \forall x.\, \alpha(x) \supset \gamma(x) \Rightarrow \Delta}$$

$$V_3: \quad \frac{\Gamma,\ \exists x.\, \alpha(x) \land \beta(x),\ \exists y.\, \beta(y) \land \gamma(y) \Rightarrow \Delta}{\Gamma,\ \exists x.\, \alpha(x) \land \beta(x),\ \exists y.\, \beta(y) \land \gamma(y),\ \exists x.\, \alpha(x) \land \gamma(x) \Rightarrow \Delta}$$

These rules are a direct implementation of "perfect" syllogism reasoning, where the order of the premises is disregarded (the antecedent of the sequent is always a set, not a sequence). Only those valid conclusions that are sanctioned by the figural effect are derived.

With the V-rules above, it can be proven that an agent will infer the valid conclusions of perfect syllogisms, but no others. Let the logic $\mathsf{B}(L_0, \tau)$ be parameterized by $L_0 = L_s$ and $\tau(S_1) = \{V_1, V_2, V_3\}$. We have the derivation:

$$\begin{array}{ll} A_K & \dfrac{[S_1]\exists x.\, Ax \land Bx,\ [S_1]\forall x.\, Bx \supset \neg Cx \Rightarrow [S_1]\exists x.\, Ax \land \neg Cx}{\begin{array}{ll} V_1 & \dfrac{\exists x.\, Ax \land Bx,\ \forall x.\, Bx \supset \neg Cx \Rightarrow_1 \exists x.\, Ax \land \neg Cx}{\exists x.\, Ax \land \neg Cx \Rightarrow_1 \exists x.\, Ax \land \neg Cx} \end{array}} \\ & \qquad\qquad\qquad\qquad \times \end{array}$$

The desired inference takes place in a single deduction step. On the other hand, for the premises (1), there is no V-rule that allows any conclusion to be derived. The sentence

$$[S_1]\exists x.\, Bx \land Ax,\ [S_1]\forall x.\, Cx \supset \neg Bx \Rightarrow [S_1]\exists x.\, Ax \land \neg Cx$$

is not a theorem of $\mathsf{B}(L_s, \tau)$.

The rules V_i are compatible with other forms of reasoning about quantified statements. One of the most common is to assign properties to particular individuals given a universal statement, as in the classic syllogism:

$$\begin{array}{c} \text{Socrates is a man} \\ \underline{\text{All men are mortal}} \\ \text{Socrates is mortal} \end{array}$$

A tableau rule to implement this inference would be:

$$V_4: \quad \frac{\Gamma,\, \forall x.\, \alpha(x) \supset \beta(x),\; \alpha(e) \Rightarrow \Delta}{\Gamma,\, \forall x.\, \alpha(x) \supset \beta(x),\; \alpha(e), \beta(e) \Rightarrow \Delta}$$

V_4 makes no additional assumptions about the interaction of syllogisms whose premises are two quantified statements.

It should not be supposed that the V-rules are being offered as a *theory* of human syllogistic reasonings. That is an ambitious task that, as we mentioned in stating the problem initially, must necessarily involve explanations of the translation between linguistic utterances and mental structures, and an elaboration of the internal mental processes that drive inferencing. The deduction model of belief makes no statement about linguistic behavior, and its modeling capabilities are descriptive rather than explanatory in any real sense. There is no claim here that humans actually use a formal internal language, or have deductive rules similar to the V-rules; Johnson-Laird [26] specifically denies that a *syntactic* deduction process could be responsible for producing the figural effect. His arguments are based on a somewhat impoverished notion of what type of processes can constitute deduction; certainly the deduction rules V_i above are an adequate descriptive model of the psychological evidence. We have shown that the deduction model can be sensitive to the *form* of belief sentences, in ways that are parallel to the behavior of human subjects doing syllogistic reasoning.

6.3 The Not-So-Wise-Man Problem

A king, wishing to know which of his three advisors is the wisest, paints a white dot on each of their foreheads, tells them there is at least one white dot, and asks them to tell him the color of their own spots. After a while the first replies that he doesn't know; the second, on hearing this, also says he doesn't know. The third then responds, "I also don't know the color of my spot; but if the second of us were wiser, I would know it."

The not-so-wise-man problem is a variation of the classic Wise Man Puzzle, which McCarthy [43] has used extensively as a test of models of knowledge. In the classic version, the third wise man figures out from the replies of the other two that his spot must be white. The "puzzle" part is to generate the reasoning employed by the third wise man. The reasoning involved is really quite complex and hinges on the ability of the wise men to reason about one anothers' beliefs. To convince themselves of this, readers who have never tried to solve it before may be interested in attempting it before reading the solution below.

The solution to the Wise Man Puzzle is as follows: *The third wise man reasons: "Suppose my spot were black. Then the second of us would know that his own spot was white, since he would know that, if it were black, the first of us would have seen two black spots and would have known his own spot's color. Since both answered that they did not know their own spot's color, my spot must be white."*

The difficulty behind this puzzle seems to lie in the nature of the third wise man's reasoning about the first two's beliefs. Not only must he pose a hypothetical situation ("Suppose my spot were black"), but he must then reason within that situation about what conclusions the second wise man would come to after hearing the first wise man's response. This in turn means that he must reason about the second wise man's reasoning about the first wise man's beliefs, as revealed by his reply to the king. Reasoning about beliefs about beliefs about beliefs ... we call reasoning about *iterated* or *nested beliefs.* It can quickly become confusing, especially when there are conditions present concerning what an agent *does not* believe. Compare:

(3)a. John believes that Sue believes some of the beekeepers are artists.
b. John believes that Sue believes that none of the artists are chemists.

(4)a. John believes that Sue believes some of the beekeepers are artists.
b. John doesn't believe that Sue believes that some of the beekeepers are not chemists.

In the case of (3), one of the correct, nontrivial conclusions that can be drawn is that

(3′) John believes that Sue believes that some of the beekeepers are not chemists.

This conclusion is not hard to draw, if one keeps in mind that it is essentially the same as the conclusion of the simple syllogism (2). An additional level of complexity has been introduced by saying that it is *John* who believes that Sue believes the premises of the syllogism. If John really believes this, it then seems obvious that he should agree that she also believes the syllogism's conclusion. Of course we could devise other complications to block this result: perhaps John is a male chauvinist and doesn't believe women can solve even simple syllogisms. But given good faith that Sue is like every other rational agent with good reasoning abilities, the conclusion (3′) seems inescapable.

A different story emerges with (4). At first glance, the valid conclusion

(4′) John doesn't believe that Sue believes that none of the artists are chemists.

doesn't seem obvious or perhaps even plausible. One way to see that it is correct, though, is to assume the opposite. If John actually believed that Sue believes *none of the artists are chemists*, it is possible to arrive at a contradiction. By the first part of (4), according to John, Sue also believes *some of the beekeepers are artists.* If John is a scholar of the psychological literature on syllogistic reasoning, then he is aware of the high probability that Sue's belief in the content of the two italicized sentences means that Sue believes *some of the beekeepers are not chemists.* And this contradicts the second part of (4), which says explicitly that John doesn't believe any such thing. The reader will note that this whole line of reasoning depends on arriving at a set of conditions that express what John believes that Sue believes, without any *not*'s occurring before the *believes.* It seems intuitively clear that reasoning about nested belief is much simpler when all the statements contain only nonnegated occurrences of *believes*, as in (3).

In the Wise Man Puzzle, nested belief contributes to the complexity of the reasoning in the same way as in (4)—to an even greater extent, in fact. The third wise man must reason about what the second wise man doesn't know (the color of his own spot); in doing this, he must also reason about the second wise man's reasoning about what the third wise man does not know (the color of *his* own spot). It is particularly annoying and troublesome to keep track of who believes what after several occurrences of not-believing in a statement of nested belief. Because human agents find it so difficult, the Wise Man Puzzle is thought to be a good test of the *competence* of any model of belief. If one can state the solution to the puzzle within the framework of Model X, so the reasoning goes, then Model X is at least good enough to show what might conceivably be concluded by agents in complicated situations involving nested beliefs.

Nested beliefs have a special importance for testing models of belief. If an agent is a rational thinker and the belief model is a good one, it makes sense to assume that the agent will use it as his own model of other agents' beliefs, and that, in turn, he will assume that these agents use the belief model as their model of other agents' beliefs, and so on *ad infinitum*. In Section 2.2.3 we called this assumption the recursion property, and we axiomatized it explicitly in the system BK.

It is possible to solve the Wise Man Puzzle within the confines of belief models that assume consequentially closed beliefs (see, *e.g.*, McCarthy [43] or Sato [61]). Such models, if they embody the recursion property, make the assumption that every agent believes other agents' beliefs are closed under logical consequence, and so on to arbitrary depths of belief nesting. While this is an accurate assumption if one is trying to model the competence of ideal agents (which is what the Wise Man Puzzle seeks to verify), it cannot represent interesting ways in which reasoning about complicated nested beliefs might fail for a less-than-ideal agent. This is the import of the not-so-wise man problem. From the reply of the third wise man, it appears that the second wise man lacks the ability to deduce all the consequences of his beliefs. The representational problem posed is to devise interesting ways in which the second wise man fails to be an ideal agent, and then show how the third wise man can represent this failure and reply as he does.

In discussing the first two problems we have already mentioned several ways in which an agent might be consequentially incomplete. The not-so-wise man problem seems to fall into the category of logical incompleteness, since the resource requirements of inference in this case are not particularly acute. We can identify at least two types (there may be more) of incompleteness that are interesting here and would be useful to represent. In one of these, the second wise man may have incomplete inferential procedures for reasoning about the other wise men's beliefs, especially if tricky combinations of *not-believing* are present, as in syllogism (4). Suppose, for instance, the second wise man were to see a black spot on the third wise man, and a white spot on the first wise man (this is the hypothetical situation set up by the third wise man in solving the classic puzzle). If he were an ideal agent, he would conclude from the first wise man's reply that his own spot must be white (by reasoning: *if mine were not white, the first of us would have seen two black spots and so claimed his own as white*). But he may fail to do this because his rules for reasoning about the beliefs of the first wise man simply are not powerful enough. For example, he might never consider assuming that his spot was black, and then asking himself what the first wise man would have said.

Another way in which the not-so-wise man might fail to draw conclusions is if he were to make an erroneous decision as to what information might be relevant to solving his problem. Although the Wise Man Puzzle has a fairly abstract setting, it is reasonable to suppose that actual agents confronted with this problem would have a fair number of extraneous beliefs that they would exclude from consideration. For example, the not-so-wise man might be privy to the castle rumor mill, and therefore believe that the first wise man was scheming to marry the King's daughter. A very large number of beliefs of this sort have no bearing on the problem at hand, but would tend to use up valuable mental resources if they were given any serious consideration. One can imagine an *unsure agent* who could never come to any negative conclusions at all, because he would keep on considering more and more possibilities for solving a problem. This agent's reasoning might proceed as follows: *I can't tell the color of my spot by looking at the other wise men. But maybe there's a mirror that shows my face. No, there's no mirror. But maybe my brother wrote the color on a slip of paper and handed it to me. No, there's no slip of paper, and my brother's in Babylon. But maybe ...*

In Section 5.2 we called the problem of specifying what beliefs an agent does not have, or does not use in solving a given task, the problem of *circumscriptive ignorance*. Without a solution to this representational problem, all agents will be modeled as unsure agents—never able to reach a conclusion about what they don't believe, even though it is obvious when the set of relevant beliefs is circumscribed.

Of course, if an agent can circumscribe his beliefs, it is possible that he will choose the wrong set of beliefs to circumscribe. The not-so-wise man may decide that the beliefs of the first wise man are not germane to the problem of figuring out his own spot's color. Thus, even though he has all the relevant information, and even sufficiently powerful inference rules and adequate resources, he may fail to come to a correct conclusion because he has circumscribed his beliefs in the wrong way. We call this type of incompleteness *relevance incompleteness*. Modeling relevance incompleteness (or having the third wise man do so) is an impossibility if it is assumed that the beliefs of agents are consequentially complete: the set of beliefs cannot be partitioned into those that are either relevant or not to a given problem; *all* the consequences of beliefs are believed.

6.3.1 Solution to the Not-So-Wise Man Problem

For this problem we use a base language L_w containing only the three primitive propositional symbols P_1, P_2, and P_3. P_i expresses the proposition that wise man S_i has a white spot on his forehead. We also require a sequence of three situations, labeled t_1, t_2, and t_3; every statement about belief will be with reference to one of these situations, and we use the theory of situations developed in Section 5.4. Finally, we need to state conditions of common belief, using the theory developed in Section 5.3; and to express the hidden assumptions of the puzzle that circumscribe the facts an agent has about his spot's color, using the circumscription operator introduced in Section 5.2. The nonintrospective logic family BK^+ defined in Section 5.5 is appropriate for this problem.

In the initial situation t_1, no one has spoken except the king, who has declared that at least one spot is white. Axioms for this situation are

$$
\begin{array}{lll}
(W1) & P_1 \wedge P_2 \wedge P_3 & \\
(W2) & [S_0, t_1](P_1 \vee P_2 \vee P_3) & \\
(W3) & (P_i \supset [S_j, t_1]P_i) \wedge [S_0, t_1](P_i \supset [S_j, t_1]P_i), & i \neq j, \quad j \neq 0 \\
(W4) & (\neg P_i \supset [S_j, t_1]\neg P_i) \wedge [S_0, t_1](\neg P_i \supset [S_j, t_1]\neg P_i), & i \neq j, \quad j \neq 0 \\
(C1) & \langle S_i : W2, W3, W4, P_j, P_k \rangle P_i \equiv [S_i, t_1]P_i, & i \neq j, k
\end{array}
$$

$W1$ describes the actual placement of the dots. $W2$ is the result of the king's utterance: it is a common belief that at least one spot is white. $W3$ and $W4$ are schemata expressing the wise men's observational abilities, including the fact that everyone is aware of each other's capabilities. $C1$ is a schema for circumscriptive ignorance. The atom $\langle S_i : W2, W3, W4, P_j, P_k\rangle P_i$ refers to the agent S_i's belief subsystem, and is true if P_i is derivable from the sentences $W2$, $W3$, $W4$, P_j, and P_k within the subsystem. The schema asserts that this is equivalent to S_i believing P_i in the initial situation, the important part of the equivalence being that if P_i *is not* derivable from just those premises, then it is not a belief. Thus the only information a wise man has about the color of his own spot is the three axioms W_2–W_4, plus his observation of the other two wise men's spots.

As an exercise of the formalism, especially the circumscription rules, let us show that all agents are ignorant of the color of their own spot in the initial situation.

$$I_2\ \frac{C_1\ \dfrac{C1 \Rightarrow \neg[S_i,t_1]P_i}{[S_i,t_1]P_i \supset \langle S_i : W2\text{–}4, P_j, P_k\rangle P_i \Rightarrow \neg[S_i,t_1]P_i}}{Circ_1\ \dfrac{\langle S_i : W2\text{–}4, P_j, P_k\rangle P_i \Rightarrow \neg[S_i,t_1]P_i}{W2\text{–}4, P_j, P_k \nvdash_{\rho(i)} P_i} \qquad N_1\ \dfrac{\Rightarrow [S_i,t_1]P_i, \neg[S_i,t_1]P_i}{\dfrac{[S_i,t_1]P_i \Rightarrow [S_i,t_1]P_i}{\times}}} \tag{6.2}$$

We have omitted some irrelevant sentences from the left side of sequents in this tableau. To show that it closes, we must pick a set of decidable deduction rules for the agent S_i so that the belief deduction operator can be evaluated. Actually, we can prove that there is no set of sound deduction rules that will enable S_i to deduce P_i from $W2$, $W3$, $W4$, P_j, and P_k, by showing that $W2\text{–}4, P_j, P_k \Rightarrow_i P_i$ is not provable in the saturated form of BK^+, which is decidable (see Section 10). This is done in the mechanically-derived proof of the Wise Man Puzzle in Appendix A.

After the first wise man has spoken, it becomes a common belief that he does not know his own spot is white. The appropriate axiom is

$$(L1)\quad [S_1,t_1]P_1 \supset [S_0,t_2][S_1,t_1]P_1 \wedge \\ \neg[S_1,t_1]P_1 \supset [S_0,t_2]\neg[S_1,t_1]P_1 \quad .$$

$L1$ is a *learning axiom*, in the sense that whatever belief S_1 has about his own spot in t_1 is learned by the others in t_2. Note that they learn about the state of S_1's beliefs as they existed in t_1. By using $L1$ and the derivation (6.2), we can prove that S_1's belief is a common belief in t_2:

$$I_2\quad \dfrac{\neg[S_1,t_1]P_1, \neg[S_1,t_1]P_1 \supset [S_0,t_2]\neg[S_1,t_1]P_1 \;\Rightarrow\; [S_0,t_2]\neg[S_1,t_1]P_1}{\underset{\times}{\neg[S_1,t_1]P_1, [S_0,t_2]\neg[S_1,t_1]P_1 \;\Rightarrow\; [S_0,t_2]\neg[S_1,t_1]P_1} \qquad \underset{\times}{\neg[S_1,t_1]P_1 \;\Rightarrow\; \neg[S_1,t_1]P_1, [S_0,t_2]\neg[S_1,t_1]P_1}}$$

With this result and that of (6.2), we can assert the following axiom about beliefs in t_2:

$$(W5)\quad \neg[S_1]P_1 \wedge [S_0]\neg[S_1]P_1$$

Given the state of common belief expressed by $W5$, we can write a new circumscriptive axiom for agents in t_2:

$$(C2)\quad \langle S_i : W2, W3, W4, W5, P_j, P_k\rangle P_i \equiv [S_i,t_2]P_i, \quad i \neq j,k$$

In t_2, all the wise men are again ignorant of their own spot's color; we could prove this fact, showing that $\vdash_{\mathsf{BK}^+} C2 \Rightarrow \neg[S_i,t_2]P_i$, in a manner similar to the proof in (6.2). S_2 relates his failure to the others, and the new situation has the additional axiom

$$(W6)\quad \neg[S_2,t_2]P_2 \wedge [S_0]\neg[S_2,t_2]P_2 \quad .$$

The third wise man at this point does have sufficient cause to claim his spot is white, but only if the second wise man is indeed wise, and the third wise man believes he is. To see how this comes about, let us prove it in the saturated form of BK^+. We will take the wise men to be powerful reasoners, and set $\tau(\nu) = \mathcal{T}_0 + A_K^+ + Circ_1 + Circ_2$, for all views ν. The sequent we wish to prove is $W1\text{–}6 \Rightarrow [S_3, t_3]P_3$.

$$C_1\ \cfrac{W1\text{–}6 \Rightarrow [S_3,t_3]P_3}{C_1\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_3 \Rightarrow [S_3,t_3]P_3}{I_2\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_3,\ P_2 \supset [S_3,t_1]P_2 \Rightarrow [S_3,t_3]P_3}{\begin{array}{c} P_2 \Rightarrow P_2 \\ \times \end{array} \qquad C_1\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_3,\ [S_3,t_1]P_2 \Rightarrow [S_3,t_3]P_3}{I_2\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_3,\ [S_3,t_1]P_2,\ P_1 \supset [S_3,t_1]P_1 \Rightarrow [S_3,t_3]P_3}{\begin{array}{c} P_1 \Rightarrow P_1 \\ \times \end{array} \qquad A_K^+\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_3,\ [S_3,t_1]P_2,\ [S_3,t_1]P_1 \Rightarrow [S_3,t_3]P_3}{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ P_2,\ P_1 \Rightarrow_3 P_3}}}}}} \tag{6.3}$$

This part of the proof is mostly bookkeeping. We have used some shortcuts, omitting obvious steps and dropping sentences from either side of the sequent if they are not going to be used.

We now must show that S_3's belief subsystem can prove P_3 from the assumptions $W2\text{–}6$ and from the belief that the other two wise men's dots are white (we are now using S_3's sequent $\Rightarrow_3$).

$$C_1\ \cfrac{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ P_2,\ P_1 \Rightarrow_3 P_3}{I_2\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ P_1 \supset [S_2,t_1]P_1 \Rightarrow_3 P_3}{\begin{array}{c} P_1 \Rightarrow_3 P_1 \\ \times \end{array} \qquad C_1\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ [S_2,t_1]P_1 \Rightarrow_3 P_3}{\begin{array}{c} I_2 \\ N_1 \end{array}\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ [S_2,t_1]P_1,\ \neg P_3 \supset [S_2,t_1]\neg P_3 \Rightarrow_3 P_3}{\cfrac{\Rightarrow_3 P_3,\ \neg P_3}{\begin{array}{c} P_3 \Rightarrow_3 P_3 \\ \times \end{array}} \qquad \begin{array}{c} N_2 \\ A_K^+ \end{array}\ \cfrac{W2\text{–}6,\ P_1,\ P_2,\ [S_2,t_1]P_1,\ [S_2,t_1]\neg P_3 \Rightarrow_3 P_3}{\cfrac{W2\text{–}6,\ P_1,\ P_2,\ [S_2,t_1]P_1,\ [S_2,t_1]\neg P_3 \Rightarrow_3 P_3,\ [S_2,t_1]P_2}{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ P_1,\ \neg P_3 \Rightarrow_{32} P_2}}}}}} \tag{6.4}$$

Note the atom P_3 on the right-hand side of the top sequent; it is equivalent to $\neg P_3$ on the left-hand side, *i.e.*, the assumption that S_3's spot is black. The sequent proof here mimics the third wise man's reasoning, *Suppose my spot were black ....* Through the observation axiom $W4$, which is a common belief, this assumption means that S_3 believes that S_2 believes $\neg P_3$. At this point, S_3 begins to reason about S_2's beliefs. Since, by $W6$, the second wise man is unaware of the color of his own spot, a contradiction will be derived if P_2 follows in S_2's belief subsystem.

$$
I_2\ \dfrac{C_1\ \dfrac{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ P_1,\ \neg P_3 \Rightarrow_{32} P_2}{W2\text{–}6,\ P_1,\ \neg P_3,\ \neg P_3 \supset [S_1,t_1]\neg P_3 \Rightarrow_{32} P_2}}{\begin{array}{cc} \begin{array}{c}\neg P_3 \Rightarrow_{32} \neg P_3\\ \times\end{array} & I_2\ \dfrac{C_1\ \dfrac{W2\text{–}6,\ P_1,\ \neg P_3,\ [S_1,t_1]\neg P_3 \Rightarrow_{32} P_2}{W2\text{–}6,\ P_1,\ \neg P_3,\ [S_1,t_1]\neg P_3,\ \neg P_2 \supset [S_1,t_1]\neg P_2 \Rightarrow_{32} P_2}}{\begin{array}{cc} N_1\ \begin{array}{c}\dfrac{\Rightarrow_{32} P_2, \neg P_2}{P_2 \Rightarrow_{32} P_2}\\ \times\end{array} & \begin{array}{c} N_2\ \dfrac{W2\text{–}6,\ P_1,\ \neg P_3,\ [S_1,t_1]\neg P_3,\ [S_1,t_1]\neg P_2 \Rightarrow_{32} P_2}{A_K^{+}\ \dfrac{W2\text{–}6,\ P_1,\ \neg P_3,\ [S_1,t_1]\neg P_3,\ [S_1,t_1]\neg P_2 \Rightarrow_{32} P_2,\ [S_1,t_1]P_1,\ [S_1]\neg P_1}{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ \neg P_2,\ \neg P_3 \Rightarrow_{321} P_1}}\end{array}\end{array}}\end{array}} \tag{6.5}
$$

S_2's reasoning (in S_3's view) takes the assumption that the third wise man's spot is black and asks what the effect would be on the first wise man S_1. Since S_1 is also ignorant of the color of his own spot, a contradiction will ensue if the first wise man can prove that his own spot is white, under the assumption $\neg P_3$. The remainder of the proof is conducted in the view 321.

$$
D_2\ \dfrac{N_2\ \dfrac{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ \neg P_2,\ \neg P_3 \Rightarrow_{321} P_1}{W2\text{–}6,\ P_1 \vee P_2 \vee P_3,\ \Rightarrow_{321} P_1,\ P_2,\ P_3}}{\begin{array}{ccc} P_1 \Rightarrow_{321} P_1,\ P_2,\ P_3 & P_2 \Rightarrow_{321} P_1,\ P_2,\ P_3 & P_3 \Rightarrow_{321} P_1,\ P_2,\ P_3\\ \times & \times & \times \end{array}} \tag{6.6}
$$

In pursuing this proof, we have assumed that the second wise man is indeed wise. There are several places in which, with slightly less powerful deduction rules for the view 32, the proof would break down. Each of these corresponds to one of the two types of incompleteness that we identified in the statement of the problem: relevance incompleteness and fundamental logical incompleteness.

Consider first the notion that S_2 is not particularly good at reasoning about what other agents do not believe, a case of fundamental logical incompleteness. One way to capture this would be to weaken the rule N_2 in the following manner:

$$N_2' : \quad \frac{\Gamma, \neg p \Rightarrow_{32} \Delta}{\Gamma \Rightarrow_{32} p, \Delta}, \quad \text{where } p \text{ contains no belief operators}$$

The modified rule N_2' would not allow deductions about what agents do not know. In particular, it would not allow the transfer of the sentence $\neg[S_1, t_1]P_1$ to the left-hand side of the sequent, a crucial step in the tableau (6.5) for the view $\Rightarrow_{32}$.

Note that the modified rule N_2' still allows deductions about what other agents do believe. For instance, if S_2 were asked whether S_1's believing P_1 followed from his believing $\neg P_2$ and $\neg P_3$, S_2 would say "yes," even with the logically incomplete rule N_2' (as in tableau (6.6) above).

A more drastic case of logical incompleteness would result if S_2 simply did not reason about the beliefs of other agents at all. In that case, one would exclude the rule A_K^+ from S_2's deduction structure. Again, the proof would not go through, because the attachment rule could not be applied in the tableau (6.5).

The notion of relevance incompleteness emerges if the not-so-wise man S_2 does not consider all the information he has available to answer the king. For example, he may think that the observations of other agents are not relevant to the determination of his own spot, since the results of those observations are not directly accessible to him. The observational axioms $W3$ and $W4$ enter into the proof tableau (6.5) in two places. Both times the rule I_2 is used to break statements of the form $p \supset [S, t]p$ into their component atoms. Preventing the decomposition of $W3$ and $W4$ effectively prevents S_2 from reasoning about the observations of other agents. A weakened version of I_2 for doing this is:

$$I_2' : \quad \frac{\Gamma, p \supset q \Rightarrow_{32} \Delta}{\Gamma \Rightarrow_{32} p, \Delta \qquad \Gamma, q \Rightarrow_{32} \Delta}, \quad \text{where } p \text{ and } q \text{ are both modal or both nonmodal.}$$

This rule is actually weaker than required for the purpose we have in mind. Consider the observation axiom $\neg P_3 \supset [S_1, t_1]\neg P_3$. There are two ways S_2 could use this axiom. If S_2 believes $\neg P_3$, he could conclude that S_1 does also. This is not the type of deduction we wish to prevent, since it means that S_2 attributes beliefs to other agents based on his own beliefs about the world. On the other hand, the axiom $\neg P_2 \supset [S_1, t_1]\neg P_2$ is used in a conceptually different fashion. Here it is the contrapositive implication: if S_1 actually does not believe $\neg P_2$, then P_2 must hold. The way this shows up in the proof tableau (6.5) is that $\neg P_3$ appears as an initial assumption on the sequent $W2$–5, P_1, $\neg P_3 \Rightarrow_{32} P_2$, while P_2 is a goal to be proved.

To capture the notion of using an implicational sentence in one direction only, we would have to complicate the deduction rules by introducing asymmetry between the left and right sides of the sequent. This is one of the major strategies used by AI rule-based theorem provers (see Section 12.3). Rather than having implicational rules of the form I_2, a typical rule-based system would use something like the following:

$$PI: \quad \frac{\Gamma, p, p \supset q \Rightarrow \Delta}{\Gamma, p, q, p \supset q \Rightarrow \Delta}$$

The implicational sentence is used in one direction only in PI, to infer q from p. If it is desired to make contrapositive inferences, then the contrapositive form of the implication must be included explicitly. Thus rules like PI allow a degree of control over the inference process to be embedded in the way the axioms are written.

In sum, we have shown that it is possible for the deduction model to represent the situation in which the not-so-wise man has less than perfect reasoning ability, preventing the third wise man from figuring out the color of his own spot. Both relevance incompleteness and fundamental logical incompleteness can be captured by using appropriate rules for $\tau(32)$.

We can also represent the second part of the problem, in which the third wise man says, *I would know my spot's color if the second wise man were wise.* To do this, we could have S_3 reason about a hypothetical individual S_2' with a complete set of rules. S_3 could then show that he would know the color of his own spot, whether S_2 responded positively or negatively about his own spot's color. In the latter case, we have already shown how the reasoning proceeds, starting at tableau (6.3). In the former case, S_3's spot would actually have to be black, and it is easy for him to prove that this is so, given S_2's response.

7. Introspection

Agents can ask questions of their belief subsystems, and, in so doing, they can form beliefs about the state of their own beliefs. Introspection about one's own beliefs is a special case of reasoning about other agents' beliefs, the crucial difference being that one has in hand an actual belief subsystem, rather than partial knowledge of its contents. In this chapter we develop a model of introspective reasoning based on the idea that a belief subsystem can issue a recursive call to itself in the course of trying to prove a statement about its own beliefs.

As was the case for belief subsystems in general, we have an interest in both descriptive and prescriptive aspects of a deduction model of introspection. Descriptive adequacy is important for modeling the behavior of agents, especially the introspective reasoning of human agents (Collins *et al.* [7] have made some important psychological studies in this area).

The structure of this chapter is as follows. First, we give an explanatory account of introspection by postulating an *introspective machine* that is an agent's model of his own beliefs. Varying amounts of self-knowledge can be represented by imposing conditions on the introspective machine. In the second section, we formalize this notion of introspection by using *introspective deduction structures* to characterize the bounds of an agent's self-beliefs. Finally, we axiomatize two special cases of the introspective model with the logic families **BS4** and **BS5**, each corresponding to a particular set of introspection conditions.

7.1 The Introspective Machine

Consider an arbitrary belief subsystem **M** of agent S. When presented with a query, **M** operates by either matching the query against its base set, or applying deduction rules to generate subqueries. During the course of this backward-chaining process, it may generate a subquery that is a question about its own state, *i.e.*, of the form $[S]p$. Such a subquery can be answered by making a recursive call to the belief subsystem again, posing the query p. Conceptually it is convenient to think of this recursive call as a call on a new belief subsystem **IM** (the *introspective machine*). The **IM** may have different characteristics than **M**—for instance, it may have a much stricter resource bound than **M**, since it is only computing a part of the answer to a larger query. A belief subsystem that relies on an introspective machine to answer self-belief queries is called an *introspective belief subsystem*.

The process of defining an introspective machine can be iterated for **IM**, producing another machine that is **IM**'s conception of *its* own beliefs, in accord with the recursion property. In what follows we will mostly be concerned with the interaction of **M** and **IM**; the ideas presented extend naturally to iterated levels, as in the example given on page 123.

IM can return one of three answers: yes, no, and undecided. A query p that is answered affirmatively by **IM** means that the agent S, upon introspecting on his own beliefs, comes to the conclusion that he believes p—that is, $[S]p$ is one of his beliefs. A negative answer, on the other hand, means that he doesn't believe his belief subsystem computes p, and so in this case $\neg[S]p$ is one of his beliefs. An undecided answer leads to no conclusion either way: neither $[S]p$ nor $\neg[S]p$ is a belief. Figure 7.1 illustrates the workings of the introspective machine by showing the way in which **M** responds to the query $[S]p$. **M** poses the subquery p to the introspective machine. If **IM** answers *yes*, then $[S]p$ is accepted as a belief, and **M**

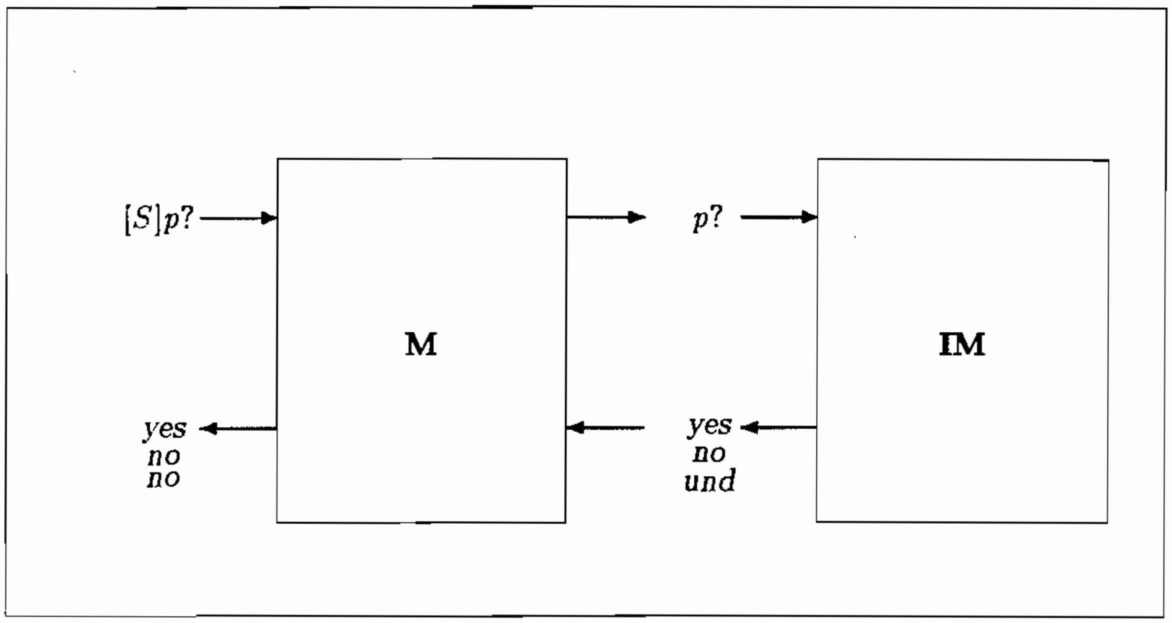

Figure 7.1 An Introspective Belief Subsystem

also answers affirmatively. If **IM** answers either *no* or *und*, **M** cannot affirm $[S]p$ as one of its beliefs, and so must return *no*.

We can summarize the response of **M** to self-belief queries or subqueries of the form $[S]p$ and $\neg[S]p$ by considering the behavior of the **IM** on p ($\mathbf{M}(p)$:*xxx* means that **M** responds *xxx* to the query p).

$$\begin{array}{lll} \mathbf{IM}(p)\text{:}\mathit{yes} & \rightarrow \mathbf{M}([S]p)\text{:}\mathit{yes}, & \mathbf{M}(\neg[S]p)\text{:}\mathit{no} \\ \mathbf{IM}(p)\text{:}\mathit{no} & \rightarrow \mathbf{M}([S]p)\text{:}\mathit{no}, & \mathbf{M}(\neg[S]p)\text{:}\mathit{yes} \\ \mathbf{IM}(p)\text{:}\mathit{und} & \rightarrow \mathbf{M}([S]p)\text{:}\mathit{no}, & \mathbf{M}(\neg[S]p)\text{:}\mathit{no} \end{array} \tag{7.1}$$

There are two important points to note about the above table. The first is that we want to leave open the possibility that an agent has no knowledge of some of his own beliefs, and this is where the answer *und* plays a crucial role. Normally, if a query p cannot be deduced within a given resource bound by **M**, then it returns the answer *no*. However, if **IM** doesn't deduce a query p, there are two distinct possibilities. In one case its associated machine **M** also would not deduce p, and **IM** would be justified in returning *no*. But it may also happen that **M** *would* derive

p, since its resource bound may be greater than than of **IM**. For example, let the query to **M** be the sentence $\alpha \wedge [S]\beta$. The control strategy of **M** might apply a rule to break this sentence into two conjunctive subqueries, α and $[S]\beta$. The solution of each subquery will consume some fractional part of **M**'s total computational resources. So although **M** may reply affirmatively to the query β posed *simpliciter*, it won't be able to derive the subquery $[S]\beta$, because the **IM** has a significantly tighter resource bound for β. In failing to derive β, the **IM** might reply *no* if it could recognize that its stricter resource bounds were not the determining factor for the failure; otherwise it might return *und*.

A second key point is that **M** may have self-beliefs by virtue of its base set of sentences, as well as by querying **IM**. This is necessary to model the sort of reasoning about self-belief that agents seem to do. To give an example (from Moore [52], pp. 5–6): an agent S might conclude that he doesn't have an older brother by reasoning, "if I had an older brother, my parents would surely have told me about it; since I have no such knowledge, it must be the case that I don't have one." Let P be the proposition that S has an older brother. We might represent the above inference by attributing the sentence $P \supset [S]P$ to S's base set. Upon introspection, the agent realizes that $\neg[S]P$ is true, and hence that (by the contrapositive $\neg[S]P \supset \neg P$) that $\neg P$ is true.

The crucial element of this piece of reasoning is the presence of a sentence involving self-belief in the base set of **M**. Since we cannot exclude such sentences, there is always the possibility that they may supersede or conflict with self-belief information from **IM**. So the cases in Table 7.1 in which **M** responds "no" to a self-belief query are correct only if the **IM** is taken to be the sole source of information about introspection. In this case, we call the introspective subsystem *intrinsic*; if there are self-beliefs generated by the base set of **M**, we call it *extrinsic*.

In terms of belief sentences about the introspective system, we can convert the responses of (7.1) into the following statements.

IM responds to p

yes	*no*	*und*
$[S][S]p$	$[S]\neg[S]p$	$\neg[S][S]p\ ^*$
$\neg[S]\neg[S]p\ ^*$	$\neg[S][S]p\ ^*$	$\neg[S]\neg[S]p\ ^*$

(7.2)

For example, if **IM** responds *und* to p, then both $[S][S]p$ and $[S]\neg[S]p$ are false of the introspective belief subsystem of agent S, *i.e.*, S has no knowledge about whether he believes p or not. The starred sentences are those that are true of intrinsic subsystems only; in extrinsic subsystems, additional information from the base sentences can override these conclusions.

An examination of the above table shows that, for intrinsic subsystems, there is no reply from **IM** for which both $[S][S]p$ and $[S]\neg[S]p$ will be true. An intrinsic introspective belief subsystem cannot have both $[S]p$ and $\neg[S]p$ as beliefs: it is consistent with respect to self-beliefs.

THEOREM 7.1. *The sentence $\neg([S][S]p \land [S]\neg[S]p)$ is true of every intrinsic introspective belief subsystem of agent S.*

Proof. By examination of (7.2) above.∎

7.1.1 Response Tables

If an introspective machine is to be an accurate reflection of its parent machine, there should be a correspondence between the responses of the two on the same query. A *response table* can be used to portray the nature of this correspondence (or lack of it). A response table consists of a column index for **M**'s answers, and a row index for **IM**'s. A cross entry (×) in the table means that the indicated responses for **M** and **IM** are simultaneously impossible for any query and any state of **M** and **IM**. For example, consider the following table:

			IM	
	p?	*yes*	*no*	*und*
M	*yes*	×		×
	no	×		×

Table 7.1 Sample Response Table

An introspective belief subsystem that conforms to this response table has the behavior that **IM** always returns *no* to every query, no matter how **M** responds (such a subsystem may not be very useful as a theory of introspection, but it would certainly be easy to implement). The cross entries in the table place strong restrictions on the linked behavior of the two machines. For instance, the fact that there is a cross in the ⟨*yes*, *yes*⟩ entry means that there is no state of **M** and its corresponding **IM** such that both **M** and **IM** would answer *yes* to a query. Note that this relationship between **M** and **IM** holds over *all possible base sets and rules of* **M**, as well as all possible queries.

Obviously, the ideal situation would be one in which the **IM** was a perfect reflection of its **M**, and so would have the following response table.

			IM	
	p?	*yes*	*no*	*und*
M	*yes*		×	×
	no	×		×

Table 7.2 Perfect Response Table

Such a *perfect* **IM** would answer *yes* and *no* exactly when its associated **M** did, and would never be undecided. However, a perfect **IM** is not always computationally realizable. So we will study **IM**'s whose responses are less than perfect, both as a feasible means of incorporating introspection into robot beliefs, and as a model of existing agents' performance. We now examine two important properties of introspective belief subsystems: *faithfulness* and *fulfillment*.

7.1.2 Faithfulness

Consider a query p that **IM** answers positively. We would expect the associated parent machine **M** to also answer *yes* if presented with p. An introspective belief subsystem for which this is true of every query p is called *positive faithful*. Similarly, if whenever **IM** responds *no* to a query, **M** also answers *no*, the subsystem is said to be *negative faithful*. A *totally faithful* subsystem is one that is both positive and negative faithful. Every definite answer returned by a totally faithful **IM** correctly reflects the answer that its associated belief subsystem would return to the same query.

The response table for positive faithfulness has a cross entry where **M** responds *no* and **IM** responds *yes*, since a positive faithful subsystem cannot exhibit this behavior. A similar response table characterizes negative faithfulness.

a. Positive Faithfulness:

			IM	
	p?	*yes*	*no*	*und*
M	*yes*			
	no	×		

b. Negative Faithfulness:

			IM	
	p?	*yes*	*no*	*und*
M	*yes*		×	
	no			

Table 7.3 Response Table for Faithfulness

Every cross entry in the response table can be used to show that a certain sentence about self-belief is true of the given introspective subsystem. For positive faithfulness, the forbidden response pair is *no* from **M** and *yes* from **IM**. A *no* answer from **M** on the query p means that p is not a belief, that is, $\neg[S]p$ holds of **M**. If the **IM** cannot answer *yes*, and the subsystem is intrinsic, then $[S][S]p$ cannot be true of **M** (see Table 7.2). This leads to the following theorem.

THEOREM 7.2. *The sentence* $[S][S]p \supset [S]p$ *is true of every positive faithful intrinsic introspective belief subsystem of agent* S.

Proof. By the argument above, $\neg[S]p$ entails that $[S][S]p$ is false for intrinsic belief subsystems. Hence $(\neg[S]p \supset \neg[S][S]p) \equiv ([S][S]p \supset [S]p)$ must hold for positive faithful intrinsic subsystems (but not necessarily for extrinsic ones).∎

For negative faithfulness, a similar argument shows that if $[S]p$ is true of **M**, $[S]\neg[S]p$ cannot hold for an intrinsic subsystem. This yields the following theorem.

THEOREM 7.3. *The sentence $[S]\neg[S]p \supset \neg[S]p$ is true of every negative faithful intrinsic introspective belief subsystem of agent S.*

7.1.3 Fulfillment

A totally faithful **IM** is guaranteed to give correct information about its associated **M** when it returns a definite answer. On the other hand, a faithful **IM** may return *und* on some queries, and so not give complete information about **M**. If the **IM** *always* responds *yes* to queries that **M** answers affirmatively, then the subsystem is called *positive fulfilled.* Similarly, if for every query that **M** answers negatively, **IM** responds *no,* it is called *negative fulfilled.* A subsystem that is both positive and negative fulfilled is perfect, since in this case **IM** responds exactly the same as **M**. Here are the response tables for positive and negative fulfilled subsystems.

a. Positive Fulfillment:

			IM	
	p?	*yes*	*no*	*undecided*
M	*yes*		×	×
	no			

b. Negative Fulfillment:

			IM	
	p?	*yes*	*no*	*undecided*
M	*yes*			
	no	×		×

Table 7.4 Response Tables for Positive and Negative Fulfillment

Examination of the response tables leads to the following theorem about fulfilled belief subsystems.

THEOREM 7.4. *For positive fulfilled belief subsystems, the sentences* $[S]p \supset [S][S]p$ *and* $[S]\neg[S]p \supset (\neg[S]p \vee [S]q)$ *are true. For negative fulfilled belief subsystems, the sentences* $\neg[S]p \supset [S]\neg[S]p$ *and* $[S][S]p \supset [S]$ *are true.*

Proof. We prove this theorem for positive fulfillment; the proof for negative fulfillment is similar. To show that $[S]p \supset [S][S]p$ is true, note that by Table 7.4(a) **IM** must respond *yes* to a query that **M** answers *yes*. Hence if $[S]p$ is true, $[S][S]p$ will be also. For the second sentence, it suffices to note that if $[S]p$ is true, $[S]p$ will be in the belief set of **M**, and hence, if $\neg[S]p$ is a belief, there will be a contradiction. Thus the sentence $[S]p \supset (\neg[S]\neg[S]p \vee [S]q)$ is true, and this is logically equivalent to $[S]\neg[S]p \supset (\neg[S]p \vee [S]q)$.

It is easy to show that fulfillment implies faithfulness. A comparison of the response tables for positive fulfillment and negative faithfulness shows that the former is a strictly stronger condition: negative faithfulness allows the **IM** to answer *undecided* when **M** responds affirmatively, which is disallowed by positive fulfillment. Similarly, negative fulfillment implies positive faithfulness.

COROLLARY 7.5. *Every positive fulfilled belief subsystem is negative faithful, and every negative fulfilled subsystem is positive faithful. A perfect subsystem is totally faithful.*

7.1.4 Summary

We can summarize the conditions of faithfulness and fulfillment for introspective belief subsystems as follows.

$$\begin{array}{rrcl} \text{positive faithfulness } (\mathit{pfa}) & \mathbf{IM}(p)\!:\!\mathit{yes} & \rightarrow & \mathbf{M}(p)\!:\!\mathit{yes} \\ \text{negative faithfulness } (\mathit{nfa}) & \mathbf{IM}(p)\!:\!\mathit{no} & \rightarrow & \mathbf{M}(p)\!:\!\mathit{no} \\ \text{positive fulfillment } (\mathit{pfu}) & \mathbf{M}(p)\!:\!\mathit{yes} & \rightarrow & \mathbf{IM}(p)\!:\!\mathit{yes} \\ \text{negative fulfillment } (\mathit{nfu}) & \mathbf{M}(p)\!:\!\mathit{no} & \rightarrow & \mathbf{IM}(p)\!:\!\mathit{no} \end{array} \tag{7.3}$$

From Corollary 7.5, there are two constraints linking fulfillment and faithfulness. There are thus nine distinct combinations of these properties that can be arranged in a lattice as shown in Figure 7.2.

The arrows indicate domination relations; the constraint *pfu*+*nfu* is thus the strongest of the possible conditions on introspective belief, in the sense that every introspective belief subsystem that obeys it also obeys every other possible combination of the faithfulness and fulfillment constraints. Indeed, the condition *pfu*+*nfu* represents an ideal, in the sense that an agent who obeyed it would have complete and accurate knowledge of his own beliefs. Note that positively fulfilled systems dominate negatively faithful ones, and negatively fulfilled systems dominate positively faithful ones.

$$\begin{tikzcd}
 & & none \arrow[dl] \arrow[dr] & & \\
 & pfa \arrow[dl] \arrow[dr] & & nfa \arrow[dl] \arrow[dr] & \\
pfu+nfa \arrow[dr] & & pfa+nfa \arrow[dl] \arrow[dr] & & nfu+pfa \arrow[dl] \\
 & pfu+pfa \arrow[dr] & & nfu+nfa \arrow[dl] & \\
 & & pfu+nfu & &
\end{tikzcd}$$

Figure 7.2 Lattice of Introspection Properties

Example. The use of introspective belief subsystems to model theories of human belief will be illustrated with one example, drawn from Hintikka [21]. He argues that if someone believes p, he also believes that he believes it (at least in the absence of strict resource limitations on reasoning). This is our condition of positive fulfillment. Hintikka goes on to argue that people will often have false ideas about their own beliefs, e.g., an utterance of the form[10]

x believes that he believes p although he does not believe it (7.4)

can be a true statement about the state of someone's beliefs. In terms of the introspective model, we would say that human belief subsystems are not positive faithful (and hence not negative fulfilled).

There is an additional curiosity to Hintikka's theory. Although the first level of introspection is characterized as being positive fulfilled but not necessarily positive faithful, it appears that subsequent levels are considered to be totally fulfilled.

[10] This is sentence 83 on page 125 of Hintikka [21].

For example, the utterance

$$\textit{x believes the following: that he believes that he believes p, although he does not believe it} \tag{7.5}$$

which is the statement of (7.4) as applied to an agent's idea of himself, is taken to be always false.

7.2 Introspective Deduction Structures

In this section we discuss the implementation of introspective machines in terms of our by-now-standard mathematical abstraction, deduction structures. As we will see, the inference rules for introspective structures have several peculiarities, which must be taken into account when defining the appropriate belief logic axiomatization in the next section. Here, we show how the conditions of faithfulness and fulfillment can be captured in a deduction structure framework.

7.2.1 *Conditions on Introspective Structures*

We introduce two introspective deduction structures, d_{ii}^{+} and d_{ii}^{-}, that are functionally related to the deduction structure d_i. The idea here is to define the boundaries of behavior of an **IM** with these structures, according to the following condition:

$$\mathbf{IM}(p) = \begin{cases} \textit{yes}, & \text{if } \mathrel{\mathsf{D}\!\!\!-}_{d_{ii}^{+}} p; \\ \textit{no}, & \text{if } \not\mathrel{\mathsf{D}\!\!\!-}_{d_{ii}^{-}} p; \\ \textit{und}, & \text{otherwise.} \end{cases} \tag{7.6}$$

By this criterion, an agent's view of his own belief subsystem must lie somewhere in the range from d_{ii}^{+} to d_{ii}^{-}; that is, he believes that his beliefs are at least those in $\mathrm{bel}(d_{ii}^{+})$, but no more than in $\mathrm{bel}(d_{ii}^{-})$. Obviously, we must have

$$\mathrm{bel}(d_{ii}^{+}) \subseteq \mathrm{bel}(d_{ii}^{-}) \tag{7.7}$$

in accord with the consistency condition of Theorem 7.1.

By imposing constraints on the relationship between d_i and its associated introspective structures, we can capture properties of the **IM** that were previously discussed. Each of the conditions on introspective belief subsystems obeying the faithfulness and fulfillment properties, as given in (7.3), can be translated into a corresponding constraint on introspective deduction structures by using Equation 7.6. For example, consider the translation of positive faithfulness.

$$\mathbf{IM}(p){:}\mathit{yes} \rightarrow \mathbf{M}(p){:}\mathit{yes} \tag{7.8}$$

The **IM** responding *yes* on p corresponds to p being in the belief set of the introspective deduction structure d_{ii}^{+}.

$$p \in \mathrm{bel}(d_{ii}^{+}) \rightarrow p \in \mathrm{bel}(d_i)$$

Since the above condition must hold for every query p, we have the following relationship between belief sets.

$$\mathrm{bel}(d_{ii}^{+}) \subseteq \mathrm{bel}(d_i)$$

In a similar manner, we arrive at the following conditions on introspective deduction structures, paralleling (7.3):

$$\begin{aligned} \mathit{pfa}: &\quad \mathrm{bel}(d_{ii}^{+}) \subseteq \mathrm{bel}(d_i) \\ \mathit{nfa}: &\quad \mathrm{bel}(d_i) \subseteq \mathrm{bel}(d_{ii}^{-}) \\ \mathit{pfu}: &\quad \mathrm{bel}(d_i) \subseteq \mathrm{bel}(d_{ii}^{+}) \\ \mathit{nfu}: &\quad \mathrm{bel}(d_{ii}^{-}) \subseteq \mathrm{bel}(d_i) \end{aligned} \tag{7.9}$$

The conditions of (7.9) must hold for arbitrary base sets of d_i. As we have assumed that the rules $\mathcal{R}^+_{ii}$ and $\mathcal{R}^-_{ii}$ of the introspective structures are fixed solely on the basis of the rules of d_i, it follows that, to enforce the introspection conditions, the introspective rules must obey the following constraints:

$$\begin{aligned} pfa: &\quad \mathcal{R}^+_{ii} \sqsubseteq \mathcal{R}_i \\ nfa: &\quad \mathcal{R}_i \sqsubseteq \mathcal{R}^-_{ii} \\ pfu: &\quad \mathcal{R}_i \sqsubseteq \mathcal{R}^+_{ii} \\ nfu: &\quad \mathcal{R}^-_{ii} \sqsubseteq \mathcal{R}_i \end{aligned} \tag{7.10}$$

In addition, the consistence condition (7.7) entails the following constraint on introspective rules:

$$\mathcal{R}^+_{ii} \sqsubseteq \mathcal{R}^-_{ii} \tag{7.11}$$

7.2.2 *Introspective Rules*

Now we turn to the question of incorporating the introspective deduction structures into rules of d_i. It should be clear that any such rules we derive will *not* be deduction rules, because they will not satisfy the criterion of provinciality. The introspective deduction structures are defined by considering the base set of d_i as a whole, not just some subset of it.

In arriving at the nonintrospective logic **BK** in Section 3.4.1, we used the following notion of derivability for deduction structures (Definition 3.8):

$\Gamma \Vdash_{\mathcal{T}} \alpha$: Let $d = \langle B, \mathcal{T} \rangle$. A sentence p is derivable from premises Γ in d if and only if $\vdash_{\mathcal{T}} \Gamma, B \Rightarrow p$.

Definition 3.8 cannot be adapted to handle reasoning about introspection. The basic reason is that its deductions all involve the sequent $\Rightarrow_i$, and the semantics of this sequent are in terms of the *truth* of sentences, without any reference to their being *beliefs* of an agent. Consider, for example, trying to define a deduction rule for positive fulfilled belief subsystems. With such a rule it should be possible to derive $[S_i]p$ from a deduction structure d_i that contains p. Given the notion of derivability expressed above, $[S_i]p$ is derivable only if the sequent $p \Rightarrow_i [S_i]p$ is provable with the new deduction rule. But this is clearly an undesirable situation; given the semantics of sequents, it justifies the inference that, whenever p is *true*, S_i believes it to be so (in the view i).

In defining introspective rules of a deduction structure, it is necessary to be able to refer to the introspective deduction structures d_{ii}^+ and d_{ii}^- during the derivation process. We now introduce two inference rules that make use of the introspective deduction structures (the *IN* rules).

$$IN_p: \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow_i [S_i]p, \Delta}{\Gamma \Vdash_{d_{ii}^+} p}$$

$$IN_n: \qquad \frac{\Sigma, [S_i]p \Rightarrow_i \Delta}{\nVdash_{d_{ii}^-} p}$$

Remarks. These are *inference* rather than deduction rules, because of the presence of the introspective deduction structures. These rules assert that the introspective machine generates self-belief sentences for **M** according to the conditions in 7.6. For example, if p is not provable in d_{ii}^-, then by IN_n it is the case that $\neg[S_i]p$ is true (recall that $[S_i]p$ on the left side of a true sequent means that it is false, hence $\neg[S_i]p$ is true). We call IN_p the *positive introspection rule*, and IN_n the *negative introspection rule.*

We now come to the main result of this section, which is to show that the theorems about faithful and fulfilled introspective belief subsystems derived by considering response tables (Theorems 7.2, 7.3, and 7.4) are also theorems of deduction structures that contain the introspective rules *IN* and obey the corresponding conditions (7.9).

THEOREM 7.6. *Let the rules for a deduction structure d_i include IN_p, IN_n, N_1, and N_2 (of $\mathcal{T}_0$). For each condition on introspective deduction structures given below, the corresponding condition on the belief set of d_i holds.*

$$\begin{array}{lrcl}
pfa{:}^* & [S_i]p \in \text{bel}(d_i) & \rightarrow & p \in \text{bel}(d_i) \\
nfa{:}^* & \neg[S_i]p \in \text{bel}(d_i) & \rightarrow & p \notin \text{bel}(d_i) \\
pfu{:} & p \in \text{bel}(d_i) & \rightarrow & [S_i]p \in \text{bel}(d_i) \\
 & \neg[S_i]p \in \text{bel}(d_i) & \rightarrow & p \notin \text{bel}(d_i) \text{ or } \forall q.\ q \in \text{bel}(d_i) \\
nfu{:} & p \notin \text{bel}(d_i) & \rightarrow & \neg[S_i]p \in \text{bel}(d_i) \\
 & [S_i]p \in \text{bel}(d_i) & \rightarrow & p \in \text{bel}(d_i)
\end{array}$$

**The asterisk on pfa and nfa indicates that the implication holds for intrinsic subsystems only.*

Proof. *pfa*: If d_i is intrinsic, then the presence of $[S_i]p$ in its belief set must arise from derivation involving an instance of IN_p:

$$IN_p \quad \frac{\Rightarrow [S_i]p}{\not\vdash_{d_{ii}^+} p}$$

The only way this tableau can close is if $p \in \text{bel}(d_{ii}^+)$. By the conditions in (7.9), this means that $p \in \text{bel}(d_i)$.

nfa: If d_i is intrinsic, then the presence of $\neg[S_i]p$ in its belief set must arise from a derivations involving an instance of IN_n:

$$\begin{array}{l} N_2 \\ IN_n \end{array} \quad \cfrac{\Rightarrow \neg[S_i]p}{\cfrac{[S_i]p \Rightarrow}{\not\vdash_{d_{ii}^-} p}}$$

The only way this tableau can close is if $p \notin \text{bel}(d_{ii}^-)$. By the conditions in (7.9), this means that $p \notin \text{bel}(d_i)$.

pfu: We prove the first implication. By (7.9), $\text{bel}(d_i) \subseteq \text{bel}(d^+_{ii})$, so if $p \in \text{bel}(d_i)$, we must have $\Vdash_{d^+_{ii}} p$. Then $[S_i]p \in \text{bel}(d_i)$ by the following proof:

$$IN_p \quad \frac{\Rightarrow [S_i]p}{\Vdash_{d^+_{ii}} p} \quad .$$

For the second implication, it suffices to note that $\text{bel}(d_i) \subseteq \text{bel}(d^+_{ii}) \subseteq \text{bel}(d^-_{ii})$ by (7.7), and hence the proof for *nfa* can be used.

nfu: We prove the first implication. By (7.9), $\text{bel}(d^-_{ii}) \subseteq \text{bel}(d_i)$, so if $p \notin \text{bel}(d_i)$, we must have $\nVdash_{d^-_{ii}} p$. Then $\neg[S_i]p \in \text{bel}(d_i)$ by the following proof:

$$IN_n \quad \frac{N_1 \quad \dfrac{\Rightarrow \neg[S_i]p}{[S_i]p \Rightarrow_i}}{\nVdash_{d^-_{ii}} p} \quad .$$

For the second implication, it suffices to note that $\text{bel}(d^+_{ii}) \subseteq \text{bel}(d^-_{ii}) \subseteq \text{bel}(d_i)$ by (7.7), and hence the proof for *pfa* can be used.∎

7.3 Belief Logics for Introspective Belief Subsystems

Having defined the introspective rules *IN* for deduction structures, we are now in a position to axiomatize these structures in belief logic families. We would like to arrive at one family for each of the different conditions on introspection presented in the lattice (7.2). Unfortunately, the language L^B is simply not expressive enough to allow this for the faithfulness conditions. Recall that faithfulness is a property whereby **M** is constrained to reply to a query in the same manner as **IM**. For instance, positive faithfulness enforces the following chain of implications:

$$\mathbf{IM}(p){:}\mathit{yes} \rightarrow \mathbf{M}(p){:}\mathit{yes} \rightarrow p \in \text{bel}(d_i) \quad .$$

The last statement, $p \in \text{bel}(d_i)$, can be expressed in L^B by asserting $[S_i]p$. However, there is no expression of L^B that captures the meaning of $\mathbf{IM}(p){:}\mathit{yes}$ in a satisfactory way when it is the antecedent of a conditional. It is not sufficient to state that

$[S_i][S_i]p$, because this simply says that $[S_i]p$ is in bel(d_i); and it could have gotten there by being in the base set of d_i, without any reference to **IM**. As a consequence, axiomatizations in the language L^{B} that are sound and complete with respect to faithful introspective subsystems all collapse into the nonintrospective family BK, and hence do not really say anything about introspective behavior.

Introspective systems that satisfy the fulfillment conditions are amenable to significant axiomatization. The chain of implication now looks like this in the positive fulfilled case:

$$\begin{array}{rcl} p \in \mathrm{bel}(d_i) & \rightarrow & \mathbf{M}(p){:}\mathit{yes} \\ & \rightarrow & \mathbf{IM}(p){:}\mathit{yes} \\ & \rightarrow & \mathbf{M}([S_i]p){:}\mathit{yes} \\ & \rightarrow & [S_i]p \in \mathrm{bel}(d_i)\ . \end{array} \tag{7.12}$$

This is a constraint on the appearance of two sentences in the belief set of d_i, and can be readily expressed in the language L^{B}. Similarly, for negative fulfillment we have:

$$\begin{array}{rcl} p \notin \mathrm{bel}(d_i) & \rightarrow & \mathbf{M}(p){:}\mathit{no} \\ & \rightarrow & \mathbf{IM}(p){:}\mathit{no} \\ & \rightarrow & \mathbf{M}(\neg[S_i]p){:}\mathit{yes} \\ & \rightarrow & \neg[S_i]p \in \mathrm{bel}(d_i)\ . \end{array} \tag{7.13}$$

As a consequence of this limitation of language, we shall consider only two introspective logics formed from L^{B}, called BS4 and and BS5 (collectively, BSn). BS4 axiomatizes positive fulfilled subsystems, and BS5 totally fulfilled ones. These are the most important logic families from the point of view of comparison with normal modal logics that have been used to axiomatize knowledge and belief (see Chapter 8).

7.3.1 *The Logic Families* BSn

The logics BSn are characterized by the presence of the rule A_{Sn}, a form of the attachment rule A that has an introspective component. Like the family BK, the families of BSn are parameterized by a base language L_0 and a tableau rule ensemble τ. The function $\tau(\nu)$ gives the tableau rules that are used by deduction structures in the view ν.

We now define two tableau rules characterizing introspective structures with fulfillment properties.

$$A_{S4}: \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, \Pi}{[S_i]\Gamma, \Gamma \Rightarrow_i \alpha}$$

$$A_{S5}: \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, [S_i]\Delta, \Pi}{[S_i]\Gamma, \Gamma \Rightarrow_i \alpha, [S_i]\alpha, [S_i]\Delta}$$

Remarks. These rules have a natural interpretation in terms of the fulfillment conditions (7.12) and (7.13). Positive fulfillment states that if Γ are believed, then so are $[S_i]\Gamma$. Thus we are justified in using the latter to derive α in each of the rules. Negative fulfillment states that if Δ are not believed, then $\neg[S_i]\Delta$ are believed. Hence we are justified in using the latter as assumptions in deriving α (recall that a sentence p on the right of the sequent sign is equivalent to the assumption $\neg p$). We might rewrite A_{S5} more suggestively as

$$A_{S5}: \qquad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, [S_i]\Delta, \Pi}{\neg[S_i]\alpha, \neg[S_i]\Delta, [S_i]\Gamma, \Gamma \Rightarrow_i \alpha}$$

We form the two versions of BSn by incorporating each of the introspective attachment rules.

DEFINITION 7.1. *The systems* BSn(L_0, τ) *are given by the following posulates:*

$\Rightarrow$ *The first-order complete rules $\mathcal{T}_0$.*

$\Rightarrow_\nu$ *Rules $\tau(\nu)$ for each nonempty view ν. A necessary condition is that Cut*is admissible in $\tau(\nu)$, and*

BS4 $\tau(ii) \sqsupseteq \tau(i)$
BS5 $\tau(ii) \simeq \tau(i)$.

A_{Sn}: A_{S4} *for* BS4, A_{S5} *for* BS5.

The first two sets of rules ($\Rightarrow$ and $\Rightarrow_\nu$) are the same as those of BK, except that conditions are placed on the nature of introspective rules $\tau(ii)$, in accord with Equation 7.10. In BS4, the relevant constraint is that, for positive fulfillment, the rules $\tau(ii)$ must subsume $\tau(i)$. For the totally fulfilled system BS5, the introspective rules must be equivalent to the original rules, so that $\tau(ii) \simeq \tau(i)$. A_{Sn} is the belief sequent form of the attachment rule A.

We now prove some theorems of these systems.

THEOREM 7.7. $\vdash_{\mathsf{BS4,BS5}} [S_i]p \Rightarrow [S_i][S_i]p$

Proof.

$$A_{S4}\ \frac{[S_i]p \Rightarrow [S_i][S_i]p}{\dfrac{[S_i]p, p \Rightarrow_i [S_i]p}{\times}}$$

∎

THEOREM 7.8. *If N_2 is admissible in $\tau(i)$, then*

$$\vdash_{\mathsf{BS4,BS5}} [S_i]\neg[S_i]p \Rightarrow \neg[S_i]p \vee [S_i]q \quad .$$

Proof.

$$\begin{array}{ll}
 & [S_i]\neg[S_i]p \Rightarrow \neg[S_i]p \vee [S_i]q \\ \cline{2-2}
D_1 & [S_i]\neg[S_i]p \Rightarrow \neg[S_i]p, [S_i]q \\ \cline{2-2}
N_1 & [S_i]p, [S_i]\neg[S_i]p \Rightarrow [S_i]q \\ \cline{2-2}
A_{S4} & [S_i]p, p, [S_i]\neg[S_i]p, \neg[S_i]p \Rightarrow_i q \\ \cline{2-2}
N_2 & [S_i]p, p, [S_i]\neg[S_i]p \Rightarrow_i q, [S_i]p \\
 & \times
\end{array}$$

∎

THEOREM 7.9. *If N_1 is admissible in $\tau(i)$, then*

$$\vdash_{\mathsf{BS5}} \neg[S_i]p \Rightarrow [S_i]\neg[S_i]p \quad .$$

Proof.

$$\begin{array}{ll}
 & \neg[S_i]p \Rightarrow [S_i]\neg[S_i]p \\ \cline{2-2}
N_2 & \Rightarrow [S_i]\neg[S_i]p, [S_i]p \\ \cline{2-2}
A_{S5} & \Rightarrow_i \neg[S_i]p, [S_i]\neg[S_i]p, [S_i]p \\ \cline{2-2}
N_1 & [S_i]p \Rightarrow_i [S_i]\neg[S_i]p, [S_i]p \\
 & \times
\end{array}$$

∎

THEOREM 7.10. $\vdash_{\mathsf{BS5}} [S_i][S_i]p \Rightarrow [S_i]p$

Proof.

$$\begin{array}{ll}
 & [S_i][S_i]p \Rightarrow [S_i]p \\ \cline{2-2}
A_{S5} & [S_i][S_i]p, [S_i]p \Rightarrow_i [S_i]p, p \\
 & \times
\end{array}$$

∎

Remarks. These theorems express the conditions on introspective deduction structures put forth in Theorem 7.6 in terms of the language of L^{B}. Note that all these are theorems of BS5, while only the first two are theorems of BS4. None of these are theorems of the nonintrospective logic BK.

The relationship between BS5, BS4, and BK can be summarized using the notion of *theory extension.*

DEFINITION 7.2. *A system* Q_1 extends a system Q_2 *if every theorem of the latter is a theorem of the former; in symbols this is* $Q_1 \rightarrow Q_2$. *A logic family* Q_1 *extends a family* Q_2 *if for every parameter* L_0 *and* τ, $\mathsf{Q}_1(L_0, \tau) \rightarrow \mathsf{Q}_2(L_0, \tau)$.

THEOREM 7.11. $\mathsf{BS5} \rightarrow \mathsf{BS4} \rightarrow \mathsf{BK}$

Proof. It should be clear that $\mathsf{BS4}(L_0, \tau)$ extends $\mathsf{BK}(L_0, \tau)$, because every proof of the latter using A_K can be duplicated in BS4 using A_{S4}. Similarly, $\mathsf{BS5}(L_0, \tau)$ must extend $\mathsf{BS4}(L_0, \tau)$, because every proof that uses A_{S4} can be duplicated using A_{S5}.∎

8. Other Formal Approaches to Belief

How does the deduction model and its logic **B** compare to other formal models and logics of belief? We examine two alternative approaches in this section:

1. Modal logics based on a Hintikka/Kripke possible-world semantics, and
2. First-order formalizations of belief, whose intended semantics is sometimes obscure.

It is possible to classify the various approaches to representing belief in terms of two independent parameters: the model of belief, and the language used to formalize the model. For the former, we distinguish symbol-processing models (of which the deduction model is one) from possible-world models. The basic difference between these is that, in the former, an agents' beliefs are characterized by the computations an agent performs on syntactic objects (symbol strings) in some internal language; in the latter, belief is taken to be a relation between an agent and abstract propositions about the world.

The second parameter of belief representation is the language of formalization. There are two that have typically been chosen: a modal language, which uses modal expressions (*e.g.*, $[S_i]p$) to represent belief; and a first-order language, which uses an ordinary predication (*e.g.*, $Bel(S_i, e)$). The *syntactic* difference between these is that, in the modal expression $[S_i]p$, p must be a sentence (or perhaps an open formula) of the language, while in $Bel(S_i, e)$, e is a term. As far as the semantics of the languages is concerned, the standard way to interpret the argument p of open formula) of the language, while in $Bel(S_i, e)$, e is a term. As far as the semantics of the languages is concerned, the standard way to interpret the argument p of a modal atom is as referring to an abstract *proposition*, while the argument e of the *Bel* predicate is taken to refer to a concrete *expression* in some language—the language of *Bel* is a *metalanguage*, and the language of e is its *object language*. If the object language and metalanguage are taken to be the same (and e thus refers to expressions in the language of *Bel*), the metalanguage is said to be *self-referential*; otherwise it is *hierarchical*.

While these are the standard interpretations of the languages with regard to belief models, other interpretations are possible. In Table 8.1, we list the various approaches by model and language type. Each column gives the systems that have been developed using a particular language type, *e.g.*, the first column lists modal systems (FO-H stands for first-order hierarchical, and FO-SR for first-order self-referential). Each row enumerates the systems developed for a given model type, *e.g.*, the first row lists systems whose intended interpretation is the possible-world model.

		Language		
		Modal	FO-H	FO-SR
Model	possible-world	*normal modal logics* Hintikka [21] McCarthy [43], [44] Sato [61] Moore [51] Levesque [38]	system SB Konolige [32]	
	symbol-processing	logic family $\mathsf{B}(L_0, \rho)$	*"syntactic" logics* McCarthy [46] Creary [8] Konolige [30] Maida [41]	Montague [49] Perlis [55]

Table 8.1 Models and Languages

As the references show, the large majority of research has been concentrated in modal logics for possible-world models, and first-order metalanguages for symbol-processing models (the so-called "syntactic" logics). Because the models and languages of these two approaches are different, comparison is difficult. Montague [49] attempted to recast normal modal logics in syntactic terms, but the resulting systems were inconsistent; his results are discussed in section 8.2.2 below. Konolige, in unpublished notes [32], formulated a first-order system SB whose models were possible worlds. This was the first research to show that there was a close correspondence between possible-world and symbol-processing semantics, because the language of SB could also be interpreted as having a symbol-processing semantics. However, because of the syntactic complexity of the first-order languages, this line of research was dropped in favor of that pursued in this thesis, namely, a modal logic for the symbol-processing model. The advantage of a modal axiomatization for comparison purposes is that the normal modal logics for possible-world semantics are relatively well-developed and understood, especially regarding the intricacies of introspection about belief. Hence it is possible to easily compare the theorems of B with these systems, and evaluate their differences and similarities.

Finally, Levesque [39] has recently used the concept of a *situation* (Barwise and Perry [2]) in developing a semantics for belief. Situations are like possible worlds in that they give truthvalues to propositions, but unlike them in that not *every* proposition need have a truth value. The model is axiomatized using a *relevance logic*. We critique this approach along with the possible-world model.

8.1 The Possible-World Model

The possible-world model of belief was initially developed by Hintikka in terms of sets of sentences he called *model sets*. Subsequent to Kripke's [36] introduction of possible worlds as a uniform semantics for various modal systems, Hintikka [22] rephrased his work in these terms. The basic idea behind this approach is that the beliefs of an agent are modeled as a set of possible worlds, namely, those that are *compatible with* his beliefs. For example, an agent who believes the sentences

Some of the artists are beekeepers.
All of the beekeepers are chemists. (8.1)

would have his beliefs represented as the set of possible worlds in which some artists are beekeepers and all beekeepers are chemists.

In formal terms, possible worlds are a collection of objects at which propositions are true or false. One possible world, w_0, is singled out as the actual world; the set of worlds compatible with an agent's beliefs are given by a binary accessibility relation R between possible worlds. Thus if world w_1 is compatible with his beliefs, we would have $w_0 R w_1$.

8.1.1 Representational Issues

In a possible world for which the sentences (8.1) are true, anything that is a valid consequence of (8.1) must also be true. There can be no possible world in which some artists are beekeepers, all beekeepers are chemists, and no artists are chemists; such a world is a logical impossibility. If beliefs are compatible with a set of possible worlds (*i.e.*, true of each such possible world), then every valid consequence of those beliefs is also compatible with the set. Thus one of the properties of the possible-world model is that an agent will believe all consequences of his beliefs—the model is consequentially closed. Hintikka, recognizing this property as a serious shortcoming of the model (his term was *logical omniscience*), claimed only that it represented an idealized condition: an agent could justifiably believe any of the logical consequences of his beliefs, although in any given situation he might have only enough cognitive resources to derive a subset of them.

The assumption of consequential closure thus causes the possible-world model to inaccurately represent the actual reasoning ability of agents: it substitutes the noncomputational, semantic notion of logical implication as an approximation to the computational processes that robot agents use. The example problems of Chapter 6 were chosen to illustrate this point; they are not handled in any framework that assumes consequential closure for agents.

We mention here one more important representational failure of the possible-world model attributable to consequential closure, namely, the problem of representing the mental state of agents as described by belief reports in a natural language.[11] Suppose, for example, the state of John's beliefs is partially given by the sentence

John believes that, given the rules of chess, White has a forced initial win. (8.2)

The statement, *given the rules of chess, White has a forced initial win* is either true in every possible world, or false in every possible world, so that either every possible world is compatible with John's beliefs, or none is. Thus (8.2) would be equivalent in the possible-world model to one of the following belief reports:

a. *John believes* **t**.
b. *John believes everything.* (8.3)

Clearly this is wrong; if it turns out that John's belief in White's forced initial win is correct, John has a good deal of information about chess, and we would not want to equate it to the tautology **t**. On the other hand, if John's belief is false and no such strategy for White exists, it is not necessarily the case that all of his beliefs about other aspects of the world are incoherent. Yet there are no possible worlds compatible with a false belief, and so *every* proposition about the world must be a belief. This particular problem stems from the possible-world treatment of belief as a relation between an agent and a proposition (*i.e.*, a set of possible worlds). All logically equivalent ways of stating the same proposition, no matter how complicated, count as a report of the same belief. By contrast, the deduction model treats belief as a relation between an agent and the *statement* of a proposition, so that two functionally different beliefs can have the same propositional content.

One interesting variation on the possible-world semantics has been developed recently by Levesque [39]. He proposes to use the *situations* of Barwise and Perry [2] instead of possible worlds. The latter can be considered a limiting case of the

[11] A good account of the relative advantages of the general symbol-processing approach to representing belief reports can be found in Moore and Hendrix [53], from which the following arguments are derived.

former: in a situation, not every proposition need be true or false. As a result, consequential closure is not a necessary property of this model, which Levesque calls *active belief*. Indeed, there are models of active belief in which p and $p \supset q$ are beliefs, but q is not. Further, it is also possible to have inconsistent beliefs without everything being believed, so that p and $\neg p$ are beliefs but q is not; or to have a belief p while not believing its logical equivalent $p \vee (q \vee \neg q)$. So the objections to the possible-world model that are based on consequential closure do not apply to the active belief model, and it certainly deserves attention as an alternative to the former. However, because of its recent origin, there are some significant omissions in the theory: problems of belief nesting and quantifying-in, for example, have not yet been addressed.

There is a further concern that is inherent in the nature of the active belief model, independent of the language used to express it. Like the possible-world model, and despite the connotations of the word "active," this model does not have any notion of computation or reasoning process, something that we have taken as basic to the concept of belief. That is, active belief allows us to state that an agent believes p and $p \supset q$ without forcing us to say that he believes q; but it does not offer any insight into conditions under which an agent might actually conclude q from the original two beliefs. Indeed, the active belief model makes predications about some conclusions; for example, if an agent believes $\neg(p \vee q)$ he must believe $\neg p \wedge \neg q$, so that it appears agents must know and use a part of DeMorgan's law. Why should this particular inference be singled out as special? It is possible to argue that there are cases, similar to those of the algebra student in the introduction or the incomplete syllogistic reasoner in Section 6.2, in which an agent would not know DeMorgan's law.

The shortcomings of a model that does not treat belief derivation as an intrinsic part of belief are brought out when we examine theories of introspection. In Chapter 7 we addressed introspection as a question of computation: how is it possible for an agent to reflect upon his beliefs, and what are the limitations of his knowledge in so doing? The computational or symbol-processing approach was fruitful in allowing us to describe and prescribe the introspective behavior of an agent.

In contrast, neither the possible-world model nor the active belief model appears to offer a felicitous framework for addressing questions of introspection. As far as the possible-world model is concerned, the accessibility relation R is a set-theoretic construct divorced from the computational notion of reflective reasoning about self-beliefs. In point of fact, it is possible to find mathematical constraints on R that correspond to two of the types of introspective behavior, namely positive and totally fulfilled systems; for example, the positive fulfillment condition is achieved if R is a transitive relation. However, these constraints carry no explanatory weight or intuitive justification from the point of view of a theory of introspection. They just happen to be the constraints that make certain sentences about the introspective behavior of an agent true, when these sentences are interpreted in a possible-world model. No theory of introspection has yet been proposed for the active belief model, but because of the nature of the model, it appears that it too must be axiomatic rather than computational in nature.

8.1.2 Normal Modal Calculi for Belief and Knowledge

Kripke [36] originally proposed possible worlds as a unifying semantics for the many different axiomatic systems of modal logic then extant.[12] Several of these calculi have been used to represent belief and knowledge. The differences between them center around whether knowledge or belief is being axiomatized, and what assumptions are made about self-belief or self-knowledge. To simplify matters, we assume that there is a single agent S, and hence confine ourselves to a single modal operator: the construction $\Box p$ is taken to have the intended meaning "the agent S believes p," $\Box\Box p$ means "S believes that he believes p," and so on.

[12] A good survey of modal calculi can be found in Hughes and Cresswell [23]; the results we cite below concerning normal modal calculi and their possible-world models come from here.

The possible-world semantics of modal systems works as follows. A model consists of a set of worlds W, a distinguished world w_0 (the actual world), a binary accessibility relation $R : W \times W$, and a boolean valuation function. The valuation function assigns a truth value to every propositional letter at every world. The value of a modal atom $\Box p$ is true at (or in) w if and only if p is true at every world accessible from w via the relation R. A sentence p is satisfied by the model if and only if it is true in w_0.

The valuation function for possible-world can be contrasted with that of a deduction structure model $\langle v_0, \phi, \mathsf{U}, D\rangle$ from Definition 4.5. Because we are dealing with a propositional language, ϕ and U are superfluous. v_0 is a valuation of propositional letters, corresponding to the valuation of propositional letters at the actual world w_0. However, here the similarity ends; the semantics of the modal atoms is completely different between the two models. In the case of possible-world models, $\Box p$ is interpreted as asserting that the *proposition* p is true at each of a set of possible worlds. In the deduction model, the *expression* $\ulcorner p \urcorner$, a syntactic object, is present in the belief set of a deduction structure. This difference in interpretation quite naturally reflects the difference in intended models.

The basic axiomatization of the possible-world model of belief (without introspection) is the modal system K:

$$\begin{array}{ll} A1. & \text{All tautologies} \\ A2. & \Box(p \supset q) \supset (\Box p \supset \Box q) \\ R1. & \textit{Modus Ponens}\text{: from } p \text{ and } p \supset q, \text{ infer } q. \\ R2. & \textit{Necessitation}\text{: from } p \text{ infer } \Box p \end{array} \tag{8.4}$$

The axioms of schema $A2$ are called the *distribution axioms*; they allow *modus ponens* to work under the scope of a belief operator. Any modal calculus that uses modus ponens and necessin, and includes all tautologies and the distribution axioms, is called a *normal modal calculus*. Normal modal calculi have the following interesting property (see Boolos [3]): if $p \supset q$ is a theorem, then so is $\Box p \supset \Box q$.

Interpreting the modal operator $\Box$ as belief, this asserts that whenever q is implied by p, an agent S who believes p will also believe q. As expected, normal modal calculi assert consequential closure when the modal operator is interpreted as belief.

The system K is sound and complete with respect to possible-world models whose accessibility relation R is unrestricted. Axioms can be added to K to characterize different properties of belief; the ones we give below have possible-world models whose accessibility relation is restricted in some way. For example, the "knowledge schema"

$KA.$ $\quad \Box p \supset p$

is added when knowledge rather than belief is under consideration (this yields the normal modal system T). KA characterizes possible-world models in which R is a reflexive relation.

We are interested here in properties of introspection; two axioms that have been used in this regard are the *positive* and *negative introspection* axiom schemata.

$PI.$ $\quad \Box p \supset \Box\Box p$
$NI.$ $\quad \neg\Box p \supset \Box\neg\Box p$

Systems with these additional axiom schemata have the following names: *weak* $S4$ (or $S4'$) is $K{+}PI$, *weak* $S5$ (or $S5'$) is $S4{+}NI$, and $S4$ and $S5$ are the weak systems with the addition of KA. $S4$ systems have transitive R, and $S5$ systems a transitive and symmetric R.

In terms of belief, positive introspection states that if an agent believes a proposition, he believes that he believes it. Negative introspection has a similar import for nonbelief: if an agent does not believe a proposition, he believes that he doesn't. Various combinations of these schemata have been proposed as a correct formalization of introspection. Hintikka [21] argues that the knowledge and beliefs of human agents satisfy PI but not NI. Moore [51] adopts Hintikka's position in his theory of knowledge and action, that is, he includes *PI*, but not *NI*. Since he also has the knowledge axiom $\Box p \supset p$, his system for a single agent is equivalent to the modal system $S4$. By contrast, Levesque [38] proposes to give an agent complete and accurate knowledge of his own beliefs, and so arrives at a system that is similar to weak $S5$.[13] Sato [61] gives Gentzen system axiomatizations for knowledge that are isomorphic in the single-agent case to T, $S4$, and $S5$, leaving it to the reader's discretion to choose his own favorite introspection conditions.

8.1.3 The Correspondence Property

It is reasonable to ask how the deduction and possible-world models compare in respects other than the assumption of consequential closure. We can phrase the comparison in this manner: in the limit of sound and complete rules for deduction structures, is the deduction model significantly different from possible-world models? Of course, the two models are composed of different entities (expressions vs. propositions), so we can always use a language that distinguishes them, having statements that are interpretable in one model and not the other. For example, if we introduce an operator that refers to sentences in the base set of a deduction structure, it be impossible to define a comparative possible-world semantics for this operator, since there are no base sets in possible worlds.

So the answer to this question of correspondence depends on the type of language used to talk about the models. Happily, we have chosen a modal language with a belief operator for the deduction logic B, and a direct comparison to normal modal logics is thus possible. Indeed, the ability to make comparisons was one of the chief motivations for choosing a modal language for B. We now assert the following unifying principle for the two approaches.

[13] Levesque uses two additional axiom schemata: a consistency schema $\Box p \supset \neg\Box\neg p$, and a schema $\Box(\Box p \supset p)$. The first schema is useful in describing agents whose beliefs can never be contradictory, because it says both p and $\neg p$ cannot simultaneously be beliefs. The second schema is a theorem of $S5$.

Correspondence Property. For every propositional modal logic of knowledge or belief based on Kripke possible-world models, there exists a corresponding deduction model logic family with an equivalent saturated logic.

The correspondence property simply says that possible-world models are indistinguishable from saturated deduction models from the point of view of propositional modal logics of belief—that is, in the limit of sound and complete deduction, the logics of B are precisely the normal modal logics of belief. To prove this claim, we will show below how each of the normal modal systems K, T, weak and strong $S4$ and $S5$ are equivalent to a saturated subfamily of **B**.

To the author's knowledge, this is the first time that the symbol-processing and possible-world approaches to belief have been shown to be comparable, in that the possible-world model is equivalent to the limiting case of a symbol-processing model with logically complete deduction. Although we state and prove the correspondence here for the case of propositional languages only, it can be extended to languages that allow quantifying-in (see Chapter 9).

In some respects the correspondence property is not a surprising result, because there is a close connection between theoremhood in a syntactic system (the deduction model) and validity with respect to a set of possible worlds. Gödel first established the completeness of a first-order system relative to first-order models: the theorems of the syntactic system are exactly the sentences valid in all models. The situation here is slightly more complicated, because the language of deduction structures contains modal operators, and the possible-world models have an accessibility relation. Yet the connection remains: for every deduction structure d whose rules are strong enough, there exists a corresponding possible-world model m such that

$$\forall w.[w_0 R w \rightarrow w \models s] \quad \text{if and only if} \quad s \in \mathrm{bel}(d) \ .$$

What we mean by "strong enough" is that the deduction structure is *saturated*: its rules are equivalent to those used by the outside observer to reason about agents' belief subsystems.

The proof of the correspondence property will be carried through on syntactic grounds. We show that each of the logics K, T, weak and strong $S4$ and $S5$ have the same theorems as a particular deductive belief logic. Table 8.2 summarizes this correspondence. Note that the knowledge versions of B are formed by the addition of the rule K_0 (see section 5.1).

	Normal Modal Calculus	Deduction Logic Family
Belief	K	BK
	weak $S4$	BS4
	weak $S5$	BS5
Knowledge	T	BK $+ K_0$
	$S4$	BS4 $+ K_0$
	$S5$	BS5 $+ K_0$

Table 8.2 Deductive Belief Logics *vs.* Normal Modal Logics

In proving the correspondence defined by the table, we will actually show only that the theorems of $S5$ are identical to those of saturated BS5 $+ K_0$, which we call $\mathsf{KS5}_s$. The remaining proofs can be derived in a similar manner. First, we prove some preliminary lemmas.

LEMMA 8.1. *The rule Cut* is an admissible rule of any of the deduction logic families above.*

Proof. The proof rests on the completeness results for these logics. Since *Cut** is propositionally sound, its addition will not destroy the soundness of a system. For concreteness, take the system BK: all theorems of BK + *Cut** are valid. Because BK is complete (Theorem 4.6), every valid sentence is a theorem of BK. Hence every theorem of BK + *Cut** is a theorem of BK, and *Cut** must be admissible.∎

The following abbreviations are used: $(\wedge\Gamma) =_{df} \gamma_1 \wedge \gamma_2 \wedge \ldots$; $(\vee\Gamma) =_{df} \gamma_1 \vee \gamma_2 \vee \ldots$; $(\wedge\Box\Gamma) =_{df} \Box\gamma_1 \wedge \Box\gamma_2 \wedge \ldots$; *etc.*

LEMMA 8.2. *Let Q be a Hilbert system containing all tautologies and the inference rule* modus ponens. *Suppose that, whenever the sentences Γ are true in a boolean valuation, the sentence p is also. Then if $\vdash_Q \gamma_i$ for all $\gamma_i \in \Gamma$, $\vdash_Q p$. p is called* a tautologous consequence *(or TC) of* Γ.

Proof. The sentence $\gamma_1 \supset (\gamma_2 \supset (\ldots \supset p))$ is a tautology, because this is equivalent to $\neg\gamma_1 \vee \neg\gamma_2 \vee \ldots \vee p$, and by assumption in every boolean valuation either one of Γ is false, or p is true. If $\vdash_Q \gamma_i$ for all $\gamma_i \in \Gamma$, then by enough applications of *modus ponens* the sentence p will be derived.∎

We will use this lemma frequently to shorten tedious proofs in Hilbert systems. For example, we will often apply the equivalences of DeMorgan's law inside complex expressions by invoking this lemma.

LEMMA 8.3. $\vdash_K (\wedge\Box\Gamma) \equiv \Box(\wedge\Gamma)$

Proof. We will prove it for $\Gamma = \{A, B\}$; the more general case is similar.

1.	$A \wedge B \supset A$	*Taut*
2.	$\Box(A \wedge B) \supset \Box A$	*1,nec,A2,modus ponens*
3.	$A \wedge B \supset B$	*Taut*
4.	$\Box(A \wedge B) \supset \Box B$	*3,nec,A2,modus ponens*
5.	$\Box(A \wedge B) \supset (\Box A \wedge \Box B)$	*2,4,TC*
6.	$\Box(A \supset (B \supset (A \wedge B)))$	*Taut,nec*
7.	$\Box A \supset \Box(B \supset (A \wedge B))$	*6,A2,modus ponens*
8.	$\Box(B \supset (A \wedge B)) \supset (\Box B \supset \Box(A \wedge B))$	*A2*
9.	$\Box A \supset (\Box B \supset \Box(A \wedge B))$	*7,8,TC*
10.	$(\Box A \wedge \Box B) \supset \Box(A \wedge B)$	*9,TC*

∎

Because Lemma 8.3 is a theorem of K, it is also a theorem of every other normal modal logic, because they all include the rules and axioms of K.

LEMMA 8.4. $\vdash_{S4'} (\wedge\Box\Gamma)) \supset \Box(\wedge(\Box\Gamma))$

Proof. We prove this lemma for the case of $\Gamma = \{A, B\}$; the more general case is similar.

1.	$\Box A \supset \Box\Box A$	*PI*
2.	$\Box B \supset \Box\Box B$	*PI*
3.	$(\Box A \wedge \Box B) \supset (\Box\Box A \wedge \Box\Box B)$	*1,2,TC*
4.	$(\Box A \wedge \Box B) \supset \Box(\Box A \wedge \Box B)$	*3,Lemma 8.3,TC*

∎

This lemma also holds for $S4$, $S5'$, and $S5$.

LEMMA 8.5. $\vdash_{S5'} (\wedge(\neg\Box\Gamma)) \supset \Box(\wedge(\neg\Box\Gamma))$

Proof. We prove this lemma for the case of $\Gamma = \{A, B\}$; the more general case is similar.

1.	$\neg\Box A \supset \Box\neg\Box A$	*NI*
2.	$\neg\Box B \supset \Box\neg\Box B$	*NI*
3.	$(\neg\Box A \wedge \neg\Box B) \supset (\Box\neg\Box A \wedge \Box\neg\Box B)$	*1,2,TC*
4.	$(\neg\Box A \wedge \neg\Box B) \supset \Box(\neg\Box A \wedge \neg\Box B)$	*3,Lemma 8.3,TC*

∎

This lemma also holds for $S5$. We now prove the main theorem of this section.

THEOREM 8.6. $\vdash_{\mathsf{KS5}_s} \Gamma \Rightarrow \Delta$ *if and only if* $\vdash_{S5} (\wedge\Gamma) \supset (\vee\Delta)$

Proof. *If* part. We show that if p is a theorem of $S5$, then $\Rightarrow p$ is a theorem of saturated **KS5**. Because the deduction logic is saturated, the rules of every view are the same. Hence we can dispense with the index signs on sequents. For clarity, we give the rules of $\mathsf{KS5}_s$ below.

1. The propositional rules $\mathcal{T}_0$.
2. $A_{S5}: \quad \dfrac{\Sigma, \Box\Gamma \Rightarrow \Box\alpha, \Box\Delta, \Pi}{\Box\Gamma, \Gamma \Rightarrow \alpha, \Box\alpha, \Box\Delta}$
3. $K_0: \quad \dfrac{\Sigma, \Box\Gamma \Rightarrow \Delta}{\Sigma, \Box\Gamma, \Gamma \Rightarrow \Delta}$

For $A1$: because $\mathcal{T}_0$ are first-order complete, every tautology is a theorem of $\mathsf{KS5}_s$.

For $A2$:

$$\begin{array}{rc} C_1 & \Box(p \supset q) \Rightarrow \Box p \supset \Box q \\ \cline{2-2} A_{S5} & \Box(p \supset q), \Box p \Rightarrow\ \supset \Box q \\ \cline{2-2} C_1 & p \supset q, p \Rightarrow q \\ \cline{2-2} & p \Rightarrow q, p \qquad q, p \Rightarrow q \\ & \times \qquad\qquad\quad \times \end{array}$$

For KA:

$$\begin{array}{rc} K_0 & \Box p \Rightarrow p \\ \cline{2-2} & \Box p, p \Rightarrow p \\ & \times \end{array}$$

PI and NI are Theorems 7.7 and 7.9, respectively.

For *modus ponens*, we show that the theorems $\Rightarrow p$ of $\mathsf{KS5}_s$ are closed under this inference. Assume that $\Rightarrow p$ and $\Rightarrow p \supset q$ have closed tableaux, and consider the following instance of *Cut**:

$$\begin{array}{c} \Rightarrow q \\ \hline \Rightarrow p \qquad p \Rightarrow q \end{array}$$

By assumption the branch $\Rightarrow p$ closes. Also by assumption, there is a closed tableau for $\Rightarrow p \supset q$, and since the only applicable rule of $\mathsf{KS5}_s$ is I_1, there must be a closed tableaux for $p \Rightarrow q$. Thus there is a closed tableaux for $\Rightarrow q$.

For *necessitation*, if $\Rightarrow p$ is a theorem, then so is $\Rightarrow \Box p$, by one application of A_{S5}. This concludes the *if* part.

Only-if part. We show that if $\Gamma \Rightarrow \Delta$ is a theorem of $\mathsf{KS5}_s$, $(\wedge\Gamma) \supset (\vee\Delta)$ is a theorem of $S5$. We do the proof here only for the case in which both Γ and Δ are nonempty; the other cases are similar.

The proof proceeds by showing how to convert every step in a block tableau of $\mathsf{KS5}_s$ into a corresponding sequence of steps for a Hilbert proof in $S5$.

Every axiom $\Gamma, p \Rightarrow p, \Delta$ of $\mathsf{KS5}_s$ is a tautology, and hence an axiom of $S5$. For the propositional rules of $\mathcal{T}_0$, the top sequent is a tautologous consequence of the bottom sequent or sequents, and for each such rule there exists a correponding sequence of Hilbert proof steps using $A1$ and *modus ponens.*

For K_0:

1.	$((\wedge\Sigma) \wedge (\wedge\Box\Gamma) \wedge (\wedge\Gamma)) \supset (\vee\Delta)$	*by assumption*
2.	$((\wedge\Sigma) \wedge (\wedge\Box\Gamma) \supset (\wedge\Box\Gamma) \wedge (\wedge\Gamma))$	*KA,TC*
3.	$((\wedge\Sigma) \wedge (\wedge\Box\Gamma) \supset (\vee\Delta)$	*1,2,TC*

Note that this proof uses only *KA* and the boolean axioms of $S5$, and so holds for any normal modal logic that includes *KA*.

For A_{S5}:

1.	$((\wedge\Box\Gamma) \wedge (\wedge\Gamma)) \supset (\alpha \vee \Box\alpha \vee (\vee\Box\Delta))$	*by assumption*
2.	$(\wedge\Gamma) \supset (\neg(\wedge\Box\Gamma) \vee \alpha \vee \Box\alpha \vee (\vee\Box\Delta))$	*1,TC*
3.	$\Box[((\wedge\Gamma) \supset (\neg(\wedge\Box\Gamma) \vee \alpha \vee \Box\alpha \vee (\vee\Box\Delta))]$	*2,nec*
4.	$\Box(\wedge\Gamma) \supset \Box(\neg(\wedge\Box\Gamma) \vee \alpha \vee \Box\alpha \vee (\vee\Box\Delta))$	*3,A2,modus ponens*
5.	$\Box(\wedge\Gamma) \supset (\Box(\wedge\Box\Gamma) \supset \Box(\alpha \vee \Box\alpha \vee (\vee\Box\Delta)))$	*4,A2,TC*
6.	$\Box(\wedge\Gamma) \supset ((\wedge\Box\Gamma) \supset \Box(\alpha \vee \Box\alpha \vee (\vee\Box\Delta)))$	*5,Lemma 8.4,TC*
7.	$(\wedge\Box\Gamma) \supset (\Box\neg\Box\alpha \supset \Box(\alpha \vee (\vee\Box\Delta)))$	*6,A2,Lemma 8.3,TC*
8.	$(\wedge\Box\Gamma) \supset (\neg\Box\alpha \supset \Box(\alpha \vee (\vee\Box\Delta)))$	*7,Lemma 8.5,TC*
9.	$(\wedge\Box\Gamma) \supset (\neg\Box\alpha \supset (\Box\neg(\vee\Box\Delta) \supset \Box\alpha))$	*8,A2,TC*
10.	$(\wedge\Box\Gamma) \supset (\neg\Box\alpha \supset (\neg(\vee\Box\Delta) \supset \Box\alpha))$	*9,Lemma 8.5,TC*
11.	$((\wedge\Sigma) \wedge (\wedge\Box\Gamma)) \supset (\Box\alpha \vee (\vee\Box\Delta) \vee (\vee\Pi))$	*10,TC*

∎

8.2 The Symbol-Processing Model

As we noted in the introduction, the symbol-processing model assumes that belief arises as a result of computation over syntactic objects. The deduction model of this thesis is a particular type of symbol-processing model, which we have chosen to formalize in a modal system **B**. There are a number of first-order formalizations of belief or knowledge in the symbol-processing tradition that have been proposed for AI systems. We have labeled these "syntactic" logics because their common characteristic is to have terms whose intended meaning is an expression of some object language. The object language is either a formal language (*e.g.*, another first-order language) as in Konolige [30], Perlis [55], and Maida [41], or an unspecified "mental" language as in McCarthy [46] and Creary [8]. In this section we will first review and respond to some general objections to the symbol-processing approach, and then contrast the deduction model and **B** with the syntactic logics.

8.2.1 Objections to the Symbol-Processing Model

The first objection is that metalanguage systems are notationally burdensome. A metalanguage contains, in addition to the normal complement of predicates and terms for individuals, a set of terms that refer to expressions in the object language. Hierarchical systems suffer from this overpopulation of terms to a greater extent than self-referential systems, because they iterate the process: the object language itself is a metalanguage for another object language. Thus the original metalanguage might have a term c that refers to an object, a term c' that refers to c, a term c'' that refers to c', and so on. Note that this objection to notation does not apply to the system **B**, which is a modal system.

One answer to this objection is that the notation, although cumbersome, has a natural interpretation. In McCarthy's [46] work, for example, c' is associated with an agent's mental concept of the object, and c'' with another agent's concept of the first agent's concept. The notation is complicated because the ideas being expressed are inherently complicated and hierarchical, and the proliferation of terms is necessary to make relevant distinctions.

Nevertheless, several authors have felt the notation to be overly burdensome, and at least two proposals have been advanced to reduce it. The first is to allow an operator that creates terms referring to objects or other terms (the η function of Konolige [30]), essentially a *syntactic sugar* device. This technique does not get rid of the hierarchy of terms, but it does clear up some of the confusion associated with which level of the hierarchy the term belongs to. A second technique, found in Maida [41], is to make the interpretation of terms dependent on the context in which they occur. However, the price to be paid is high: in standard first-order semantics the interpretation of terms is always independent of the construction they appear in. The nonstandard semantics of this approach has not been developed sufficiently to consider it seriously as an alternative representation for belief: why abandon all the results of first-order logic to achieve a single notational convenience?

An interesting objection to symbol-processing models in general has been raised by Levesque [39]. He considers a belief report of the form

$$\textit{It is believed that either } \alpha \textit{ or } \beta \textit{ is true,} \tag{8.5}$$

and notes that the order of α and β seems to be completely irrelevant to the meaning of the report. Yet in the syntactic approach, he continues, this report must be represented as either $Bel(S, \ulcorner \alpha \vee \beta \urcorner)$ or $Bel(S, \ulcorner \beta \vee \alpha \urcorner)$. Presumably, these represent different belief sets, one containing the sentence $\alpha \vee \beta$ and the other its reverse. Levesque's point is that the syntactic approach makes the left-right order of disjuncts "semantically significant," whereas, from the belief report evidence, it appears that order is immaterial.

On the face of it this objection seems plausible, but it fails to hold upon closer examination because its premises are false. First, consider that the whole foundation of a symbol-processing approach is that a proposition's *form* of expression is an important part of belief; we presented several arguments at the beginning of this chapter as to why this is so. The important point here is that not all logically equivalent ways of stating a proposition count as the same belief, because the computations that operate on expressions may treat them differently. To take an example from AI planning systems, the sentences $\alpha \supset \beta$ and $\neg\alpha \vee \beta$ are often treated differently by inferential mechanisms, most notably in rule-based deduction systems (see Section 12.3).

If the form of expression is important, there might indeed be a distinction between the two sentences $\alpha \vee \beta$ and $\beta \vee \alpha$ in a belief set. Then again, there may not; it depends on the property of the particular inference mechanisms employed in the belief subsystem. For example, the tableau rules $\mathcal{T}_0$ make no distinction between the two forms, and anything derivable with one will be derivable with the other. So one answer to the objection is that the belief subsystem of agents is such that either disjunct is equivalent to the other. Levesque calls this answer "semantically unmotivated," but it is not clear what he means. Certainly the semantic picture presented is perfectly adequate: belief involves inference over sentences, and certain sentences fall into the same equivalence class for these inferences. What Levesque perhaps means by this phrase is that there is no semantic account that uses possible worlds or some similar set-theoretic construction; but this certainly isn't a failing of the symbol-processing model, which has no use for this type of semantical machinery.

Finally, there seems to be a confusion between belief *reports* and the expressions that actually are present in belief sets (an easy confusion to fall into, and one which we were careful to point out in discussing the syllogism problem in Chapter 6). Why is it necessary that the "or" of belief reports in English correspond to the logical symbol "$\vee$" of a first-order language? On linguistic grounds the evidence is very much against such a simple identification. A different internal language for the agent might fare better in translation; after all, nothing in the symbol-processing model forces us to use the standard first-order connectives, and we are free to pick ones that might be in better agreement with the assumed actual internal language of agents. For example, the internal language might contain a symbol "$\sqcup$" whose single argument was an unordered set of two elements. Then the inclusion of $\sqcup(\{\alpha, \beta\})$ in the belief set would be the single way of expressing the meaning of (8.5). The point is that the internal language might have only a single such mode of expression, but the metalanguage itself might have many ways of writing this expression; we might even intend that the metalanguage expressions $\ulcorner\alpha \vee \beta\urcorner$ and $\ulcorner\alpha \vee \beta\urcorner$ *mean* $\sqcup(\{\alpha, \beta\})$. Something along these lines has already been proposed by McCarthy [42] in his *abstract syntax* proposal for the metalanguage.

8.2.2 Self-Referential Languages

A notion that has intrigued logicians since the inception of the Hilbert program is the possibility of proving the consistency of first-order logic within its own framework (see Kleene [28], pp. 247ff.). To do this, it is necessary to have terms in the language that refer to expressions of the language. A method that has become standard is to embed a theory of arithmetic in the language, and then assign expressions of the language to unique numbers. We call a language of this sort *self-referential.* It should be carefully noted that the language of B is *not* self-referential. Even though we have stated that agents and the outside observer use the same language, by this we simply mean that the language of the agents and the outside observer use the same collection of primitive symbols and have the same rules of formation. The sequence of languages in B is strictly hierarchical: the observer can refer to expressions in agents' belief subsystems, and the agents can refer to expressions in their model of other agents belief subsystems, and so on; but a language in the hierarchcy never refers to expressions in a language above it in the hierarchy.

The motivation for using a self-referential language to axiomatize belief is twofold. First, there is an apparent lessening of the notational burden associated with hierarchical languages. There is no need to have an infinite chain of language/metalanguage pairs, each having terms referring to expressions in their predecessor; all terms refer either to object or to expressions in the same language. Offsetting this syntactic simplicity is an added referential complexity relative to hierarchical systems. Terms that can refer to themselves are often hard to understand, and intuitions about their consequences are prone to be misleading or wrong.

In this matter, as in many others about notational convenience, individual choice seems to be the best arbiter.

A second more crucial motivation is that self-referential languages are more expressive than hierarchical languages (the latter having no self-referencing terms), and this difference might be important in representing belief. However, the evidence to make this case is less than overwhelming. Perlis [56] cites the example of the belief report

$$S \textit{ has no religious beliefs} \tag{8.6}$$

as not being representable using a hierarchical language. This is simply false; the expression $\forall x.[\mathit{Rel}(x) \supset \neg \mathit{Bel}(S, x)]$ will do nicely in the hierarchical system of Konolige [30], where *Rel* is a predicate whose intended meaning is that its argument contains significant reference to religious objects.[14]

On the other hand, there is good reason to assume at least a partially hierarchical approach. Presumably an agent's internal language would not contain any references to the outside observer's language, which we assume to be completely removed from the environment the agent operates in. Hence there is no need for the observer's language, the language of B, to be self-referential. As far as the agent himself is concerned, we have shown in the first half of this thesis how an agent can model other agents, including himself, by assuming that they also have belief subsystems similar to his own. In no part of this endeavor were we forced to assume that the language was self-referential; indeed, in the place where we might most expect it, namely in dealing with introspection, we arrived at a powerful and natural theory without using self-referential statements.

[14] By "significant" reference we mean to exclude such essential nonreligous beliefs as the tautology "either there is a God or there isn't a God."

Even if, for whatever reasons, a self-referential approach is taken, there are considerable axiomatic difficulties that must be overcome. Self-referential languages prove peculiarly prone to inconsistency when intuitive axiomatizations of various concepts are assumed. Gödel was the first to show such an inconsistency: a first-order self-referential system could correctly express its own consistency only on pain of being inconsistent. And Tarski proved a similar result for the truth-conditions of a self-referential language: if the desired axioms about truth were added to the system, it would be inconsistent. A more recent result along these lines that bears more directly on the present research comes from Montague [49], who undertook an investigation of modal concepts using a self-referential language. His method was to introduce a predicate N (for *necessity*), similar to *Bel*, whose argument is a term referring to an expression. For any modal system, it is possible to form a corresponding system using N by simply writing the axiom schemata of the modal system using N and terms referring to the appropriate sentences; for example, KA would be expressed as $N(\ulcorner\phi\urcorner)\supset\phi$, where $\ulcorner\phi\urcorner$ is a numeral referring to the expression ϕ. Montague showed that the resulting first-order systems were inconsistent.

Although Montague was interested in the interpretation of N as necessity rather than belief, his results can be applied to the standard modal systems discussed above in Section 8.1.2. In particular, the system T (the simplest normal modal logic for knowledge) is inconsistent when expressed in a self-referential language, and hence so are $S4$ and $S5$. Montague's results do not imply that the weaker systems K, $S4'$ and $S5'$, which omit KA, are inconsistent, and their consistency is still an open question. Nevertheless his results are discouraging for those who want to employ self-referential languages, because often it is desirable to consider a formalization in which all beliefs are assumed to be true: for example, it is a useful simplification in solving the Wise Man Puzzle and other puzzles of belief.

Perlis [55], in an attempt to recover from the negative results of Tarski and Montague, has recently considered self-referential languages that use a modified truth-schema (first defined by Gilmore [15]) of the form

$$T(\ulcorner\phi\urcorner) \equiv \phi^* \,, \tag{8.7}$$

where ϕ^* is ϕ with all occurrences of $\neg T(\ulcorner\psi\urcorner)$ replaced by $T(\ulcorner\neg\psi\urcorner)$. The resulting system can be shown to be consistent, whereas replacing ϕ^* with ϕ in (8.7) is Tarski's schema, which is inconsistent.

It is not clear what the relevance of this truth-schema is to the representation of belief. Perlis ([55], p. 11) claims that it makes the usual hierarchical constructions for belief unnecessary , but this is not obvious *prima facie*, and the evidence to back this claim is lacking. One immediate objection that comes to mind is that even though some version of truth can be consistently axiomatized using Gilmore's schema, Montague's results still hold for the necessity (or belief) operator N. Presumably, the relevant modification would be to change the axiom KA to $N(\ulcorner\phi\urcorner) \supset \phi^*$, but the consistency of Montague's axioms for necessity does not follow from the consistency of the simple truth-schema (8.7), and remains an open question.

A more subtle objection is that even if the consistency of a self-referential axiomatization of belief à *la* Gilmore can be shown, there remains the problem of explaining exactly what the starred construction ϕ^* means, and what its representational consequences are. The use of the $*$ operator would replace some occurrences of $\neg Bel(S, \ulcorner\phi\urcorner)$ with $Bel(S, \ulcorner\neg\phi\urcorner)$. This is a syntactic manipulation, and we might ask what its semantic consequences are. Is consistency achieved here only at the expense of throwing away most of the representational advantage *vis*-a-*vis* hierarchical systems that was originally expected? If it is, then the motivation for using a self-referential language in the first place is undermined.

8.2.3 Syntactic Systems

In this section we review current syntactic approaches (both hierarchical and self-referential), and compare them to the deduction model.

McCarthy [46] has presented some incomplete work in which *individual concepts* are reified in a first-order logic. Exactly what these concepts are is left deliberately vague, but on one reading they can be taken for the internal mental language of a symbol-processing cognitive framework. He shows how the use of such concepts can solve standard representational problems of knowledge and belief addressed by the possible-world model, e.g., quantification into belief contexts.

A system that takes seriously the idea that agent's beliefs can be modeled as a theory in some first-order language is proposed by Konolige [30]. A first-order metalanguage is used to axiomatize the provability relation of the object language. To account for nested beliefs, the agent's object language is itself viewed as a metalanguage for another object language, and so on, thereby creating a hierarchy of metalanguage/object language pairs.

Perlis [55] presents a more psychologically oriented first-order theory that contains axioms about long- and short-term memory. The ontology is that of an internal mental language.

These axiomatic approaches are marred by two defects: lack of a coherent formal model of belief, and computational inefficiency. Regarding the first one: the vagueness of the intended model often makes it difficult to claim that the given axioms are the correct ones; there is no formal mathematical model that is being axiomatized.[15] In arriving at the deduction model of belief, we have tried to be very clear about what assumptions were being made in abstracting the model, how the model could fail to portray belief subsystems accurately, and so on. In contrast, the

[15] To some extent this criticism is not applicable to the formalism of Konolige in [30], because here the intended belief model is explicitly stated to be a first-order theory.

restrictions these syntactic systems place on belief subsystems are often obscure. What type of reasoning processes operate to produce consequences of beliefs? How are these processes invoked? What is the interaction of the belief subsystem with other parts of the cognitive model? These types of questions are often begged by simply writing axioms and then trying to convey an intuitive idea of their intended content.

A second shortcoming is that efficiently mechanizable automatic proof methods for the syntactic axiomatizations are not provided. As we have mentioned, a system that uses standard theorem-proving techniques on axioms about belief can run into severe computational problems. Many of the assumptions that were made in arriving at the deduction model were based on technical convenience and deductive efficiency, the main one being the closure property. The end result is a simple rule of inference, the attachment rule A, that has computationally attractive realizations. On the other hand, formalizations that try to account for complex procedural interactions (as in Perlis' [55] theory of long- and short-term memory), or that use a metalanguage to simulate a proof procedure at the object language level (as in Konolige [30]), have no obvious computationally efficient implementation.

Finally, the syntactic logics for belief have not been shown to be in conformity with the *correspondence property*: the notion that, in some ideal limit, the properties of these logics should be the same as those of logics based on the possible-world model. The logic B actually does comply with this principle, and forms a link between syntactic and possible-world formalizations of belief.

9. Quantifying In

Often we will want to say that an agent believes of an individual that he has a certain property, without saying who that individual is. For example, the sentence

$$\textit{there is a man whom John believes to be a spy} \tag{9.1}$$

states that *John* believes a particular individual is a spy, but *we* do not know which one. In terms of the deduction model, we would characterize John's mental state as one in which the sentence $Spy(\eta)$ was contained in his belief subsystem, where η is the name of the individual in question. There is no particular sentence that we know to be in the belief subsystem, just one whose form is $Spy(\eta)$.

The language L^{B} is not sufficiently expressive to represent the partial information about John's belief state contained in (9.1). This is because the sentences that appear as arguments to the modal operators cannot have any unspecified parts—they are always closed sentences, and refer to themselves. What we really want to do is to say that there is some unspecified individual whose name is part of a sentence that John believes. In the notation of B, this would be something like

$$\exists x.\ [John]Spy(\ulcorner x \urcorner) \tag{9.2}$$

where the quotes around x indicate that we mean to denote the *name* of the individual x. Because x appears both inside and outside of a modal context, expressions like (9.2) are referred to as having a *quantified-in* form. Expressions that are quantified-in are not sentences of L^{B}.

In this chapter we introduce the logic family **qB** whose language is an extension of B's that permits quantifying in. The axiomatization of **qB** is similar to that of B, and in particular the attachment rule suffers only minor modification. The models of **qB** will also be the same as those of B, and we will prove theorems like those proven for B in Chapters 3 and 4, including soundness and completeness results.

9.1 The Language of qB

The language L^{qB} of **qB** is defined in a manner similar to the language L^{B} in Definition 3.6.

DEFINITION 9.1. *Let* $\{S_0, S_1 \ldots\}$ *be a countable set of agents, and* L_0 *a first-order base language (as given in Definition 3.1). A sentence of* L^{qB} *based on* L_0 *is defined recursively by the following rules.*

1. *All formation rules of* L_0 *are formation rules of* L^{qB}.
2. *If* ϕ *is a formula of* L^{qB}, *then* $[S_i]\phi$ *is also a formula.*

The language L^{qB} *reserves three parts of* L_0 *for special purposes.*

1. *A countable subset of the constants of* L_0, *which are called* identifiable constants *or* id constants.
2. *For each constant* a *of* L_0, *a constant* $a^{\bullet}$ *called its* bullet constant. *In sentences of* L^{qB}, *bullet constants may only appear in the context of a modal operator.*
3. *A predicate* I_i *of one argument for each agent.*

Id constants are written as $\hat{c}_i$. *Individual constants that are not id constants or bullet constants are called* unspecific constants.

The difference in the definition of L^{B} and L^{qB} is in the second clause: for L^{qB}, we allow ϕ appearing in the context of a modal atom to be a *formula* with free variables, rather than just a closed sentence. Hence L^{qB} allows "quantified-in" sentences, for example,

$$\begin{array}{l} \exists x.\, [S_i]Px \\ \forall x.\, Px \supset [S_i]Qx \quad , \end{array} \tag{9.3}$$

which are not sentences of B.

9.1.1 *Identifiable Constants*

In the language L^{qB}, we can no longer interpret the arguments of modal atoms as sentences that stand for themselves, because they may contain variables that have been quantified outside the context of the modal atom. There is a large difference, for example, between the sentences

$$\exists x.\, [S_i]Px \tag{9.4}$$

and

$$[S_i](\exists x.\, Px) \quad . \tag{9.5}$$

The second sentence (which is a sentence of L^{B} as well), states that the expression $\exists x.\, Px$ is contained in S_i's belief subsystem. The first states that there is an expression of the *form* $P\eta$ in his belief subsystem, but that the exact expression is dependent on the individual that is the referent of x. The formula 'Px' inside the modal operator, with the free variable x, is thus a *schema* that stands for an expression constructed out of the predicate symbol P and a constant term naming (for S_i) the individual referred to by x. Now, not just any term will do here. We need a term that *identifies* the individual x for the agent S_i. What we mean by "identifies" is problematic, and depends on the problem domain under consideration. For example, S_i may believe that Noah's ark was XXX cubits long without being able to construct a similar ark, because he does not know how long a cubit is. The term "XXX cubits" does not identify a particular distance measure for this agent. On the other hand, if he believes that the ark was 120 feet long, then he would be able to use a tape measure to lay out the beams and ribs and so on, so that the ark he builds is the correct length. The term "120 feet" has a distinguished computational meaning for the cognitive system of S_i. Its presence in his belief subsystem enables his other subsystems to carry out actions that implement his goals, whereas "XXX cubits" does not.

Terms that have this special cognitive significance will be called *identifiable constants*, or more simply *id constants.* They are a distinguished subset of the individual constants of the language L^{qB}. What actually counts as an id constant depends on the particular characteristics of the intended domain. But, in almost any domain, two particular types of id constants are almost certain to be present: those that are *standard names* for individuals, and those that are *agent-relative.* Moore [51] illustrates the standard name concept with an example from an AI system that analyzes electronic circuits (Stallman and Sussman, [64]). Identifiers are assigned to components of the circuit, so that a transistor might be given the label $Q301$. $Q301$ is a standard name that is known to both the system and its users, and can be used to represent facts about that particular transistor; for example, the system believes that the transistor is burned out if it believes that $Q301$ is burned out.

Standard names are a very important mechanism for holding beliefs about particular objects, but certainly not the only one. Another equally important class of names are those that are agent-relative. By this we mean that the names are descriptions that refer to an object when they are evaluated relative to an agent. The best example is a term that has the effect of the pronoun "I," that is, it always refers to the agent itself. Other types of agent-relative descriptions are sure to be important in domains that require a robot agent to perform actions. For example, a robot might have a description of an object as "the table six feet in front of me;" this description is an identifiable constant for the purpose of moving to the table. Agent-relative names are, in general, not standard names, as this example illustrates. Although "$Q301$" is a sufficient description for referring to a particular transistor in any context, "the table six feet in front of me" is not. In fact, it may pick out the wrong table, or no table at all, when used by a different agent.

In formal terms, an id constant can be thought of as a special name for an individual in the language L^{qB}; these names are reserved beforehand when we define the language. A model of L^{qB} specifies a *naming map* from individuals to id constants. This mapping may be only partial, in two senses: there may be individuals for which no id constant is available, and there may be id constants that are not names for any individual. However, the mapping is a function—only one id constant can be associated with an individual. The idea is that in the real world there are many individuals, and an agent has beliefs about only a subset of these; hence the incomplete mapping from individuals to id constants. An agent may also have beliefs about what he thinks are real individuals, but which are in fact nonexistent; and so there can be id constants that are not the names of actual individuals. The distinguished predicate I_i picks out those individuals that actually have associated id constants for the agent S_i.

In terms of quantified-in sentences of the language L^{qB}, id constants have the following effect. If $\exists x.[S_i]Px$ is true, then there is an associated id constant $\hat{c}$ given by the naming map such that the sentence $P\hat{c}$ is in S_i's belief subsystem (and thus, $I_i x$ is true of the individual x). On the other hand, the converse does not necessarily hold: it may be the case that there is a sentence $P\hat{c}$ in S_i's belief subsystem (with id constant $\hat{c}$), but $\exists x.[S_i]Px$ cannot be asserted because there is no actual individual whose identifiable name is $\hat{c}$. Variables that are quantified outside the scope of modal operators always range over the set of actual individuals.

It is interesting to compare the semantics of quantified-in variables given here with typical possible-world accounts. Hintikka [21] noted that it is possible to interpret sentences of the form $\exists x.[S_i]Px$ as asserting that S_i believes the property P holds of an individual x who is the same in every possible world compatible with S_i's beliefs. Of course, this presupposes that we have an account of properties and individuals in possible worlds, and so the semantics given in Section 8.1.2 must be extended. Kripke [36] gives a quantificational account of possible worlds, assuming that at each world w there is an arbitrary universe U_w of individuals, and a denotation function mapping terms to members of this universe. For each n-place predicate letter P^n, w contains the extension of P^n. Thus each possible world specifies a valuation for all ordinary atoms. In the normal manner, this valuation is extended to modal atoms by reference to the accessibility relation R; from these, the valuation of any sentence at the world can be determined.

If we apply Hintikka's interpretation of quantifying-in to this model, then $\exists x.[S_i]Px$ asserts that there is some individual k in U_{w_0} (the actual universe) such that Pk is true in every world w compatible (for S_i) with w_0. That is, the quantified-in variable x picks out the same k in each compatible possible world. Now suppose there is some constant c in the language that refers to k in every possible world; such constants are called *rigid designators* by Kripke [37]. The existential sentence is then equivalent to the sentence $[S_i]Pc$, in which the quantified-in variable is replaced by a rigid designator. Following Moore [51], it seems natural to identify rigid designators with standard names, because the denotation of the latter is a fact of language: no matter what possible world it is interpreted in, a standard name always refers to the same individual.

This account of quantifying-in now appears much closer to the interpretation given for L^{qB} using id constants. In the latter, $\exists x.[S_i]Px$ entails that there is some id constant $\hat{c}$ such that $[S_i]P\hat{c}$ is true; when $\hat{c}$ is a standard name, this is exactly the same as the possible-world analysis. Of course, there are other ways of believing something about a particular individual than having a standard name for the individual; we mentioned agent-relative names, for instance. In this regard, the deduction model appears to be more flexible in the type of id constants for which it will sanction an interpretation. The deduction model allows us to talk about terms of a particular sort in an agent's belief subsystem, terms that may have a special performative influence on the cognitive structure of an agent. On the other hand, the denotation function associated with a possible world is specified independently from the accessibility relation, and so cannot take into account any agent-relative interpretation. In point of fact, nothing in the possible-world model prevents a single possible world from being compatible with the beliefs of several different agents. In such a world, any denotation of an agent-relative term such as "I" would appear to do violence to our intuitions about reference.

A cautionary note must be added to this argument, because the issues of reference are exceedingly subtle, and we have not yet had any experience in exercising the deduction model formalism in an actual robot agent; any claim for greater expressive power of quantifying-in in the deduction model must await further evidence to refine the arguments presented. One established result is presented at the end of this chapter, however. The language L^{qB} has an interpretation in the Kripkean possible-world model to represent belief, and, when the system qB is restricted in the appropriate way, it is sound and complete with respect to these models. Thus the correspondence property holds for qB.

9.1.2 Bullet Constants

In this subsection we introduce the technical device of *bullet constants* for dealing with unknown id constants.

Consider the rule of existential generalization (E_1 of L_0). Applying it to sentence (9.4) yields the tableau

$$E_1 \quad \frac{\Rightarrow \exists x.\, [S_i]Px}{\Rightarrow [S_i]Pa} \tag{9.6}$$

in which the sentence Pa appears as the argument to the modal operator. However, this conflicts with our interpretation of Px as a schema in which x must be replaced by an id constant. An initially plausible solution is to specify that the substitution constant a be an id constant. This fails because our semantics forces a to be a particular id constant; an arbitrary id constant may not even be the identifiable name of a real individual. In substituting a constant a for the quantified-in variable x, we require a means of specifying the id constant associated with the *referent* of x. This is the purpose of bullet constants. Bullet constant are used only in modal constants, and like free variables in these contexts, refer to id constants; thus the bullet constant $a^{\bullet}$ picks out the id constant associated with the referent of a. Bullet constants are a technical device for keeping track of the origin of id constants generated by substitution of variables in the course of a tableau proof.

We now define the substitution operation for sentences of L^{qB}. The only difference from Definition 3.2 is in quantified-in atoms.

DEFINITION 9.2. *Let α be a formula of L^{qB}. For every variable x and individual constant a the formula α^x_a is given inductively by the rules (1), (2), and (3) of Definition 3.2, together with the rule*

$$4. \qquad ([S_i]\alpha)^x_a = \begin{cases} [S_i]\alpha, & \text{if } x \text{ does not appear free in } \alpha; \\ [S_i]\alpha^x_a, & \text{if } a \text{ is a bullet constant;} \\ I_i a \wedge [S_i]\alpha^x_{a^{\bullet}}, & \text{otherwise.} \end{cases} .$$

Note that the bullet constant $a^{\bullet}$ always picks out *some* id constant; the insertion of the conjunct $I_i a$ is necessary to ensure that the referent of a actually has an associated id constant in the naming map. An example of substitution is

$$(Px \wedge [S_i]Px)^x_a = Pa \wedge I_i a \wedge [S_i]Pa^{\bullet}. \tag{9.7}$$

Note that the intended semantics of the two sentences is the same. The one on the left asserts that property P holds of an individual, that S_i has a belief sentence predicating P of the id constant associated with the individual, and that the individual has the name a. The right-hand sentence asserts directly that the individual named a has the property P, that this individual has an associated id constant, and that S_i has a belief sentence predicating P of the id constant. The right-hand side is *not* equivalent to the sentence $Pa \wedge [S_i]Pa^{\bullet}$, because there is nothing in the latter qualifying $a^{\bullet}$ as the id constant associated with the individual named a (for example, S_i may not believe such an individual exists, in which case $I_i a$ is false).

One unfortunate consequence of the complex substitution operation is that we must be more careful in using predicate variables in proofs. Normally, when we prove a "theorem" such as $p \supset p$, we have really constructed a proof schema in which the variable p can be replaced by any sentence, and the result will be a theorem. Although this will always be true of substitution for *sentence* variables, there are difficulties in applying substitution to open formulas. Consider, for example, the schema $\forall y.A(y) \supset \forall x.I_i x$, where $A(x)$ stands for a formula with free variable x. If we substitute $[S_i]Py$, then a valid sentence is obtained; if we subsitute $[S_j]Py$ or just Py, the sentence is falsifiable. When using open schemata of this sort, it is important to check carefully for any restrictions on substitutions.

9.1.3 Schematic Constants

The definition of substitution instances allows bullet constants to appear in the argument to belief operators, as in (9.7). By the informal semantics we have given,. a bullet constant $a^{\bullet}$ picks out an id constant of L^{qB} associated with the individual referred to by a. However, both the referent of a and the naming map are not fixed by the language, and so we do not know exactly which id constant $a^{\bullet}$ specifies. This causes problems for the attachment rule, which relates belief operators to deduction in an agent's belief subsystem. For example, how can we show that $[S_i]Pa^{\bullet}$ is valid, when we do not know exactly which sentence $Pa^{\bullet}$ refers to?

One answer is that $[S_i]Pa^{\bullet}$ will be valid if for *every* id constant $\hat{c}_j$, there is a proof of $P\hat{c}_j$ for S_i. It might be possible to axiomatize qB using this fact, where the attachment rule for $[S_i]Pa^\bullet$ asked for the class of derivations $\Vdash_{\rho(i)} P\hat{c}_j$ for each $\hat{c}_j$. However, there are two problems with this approach. The most obvious one is that it does not generate very efficient proof methods, since one must show a class of belief derivations exist in order to find proofs of sentences involving modal atoms. A second and less obvious complication is that it is difficult, if not impossible, to find an axiomatization that is complete.

Instead of trying to prove the class of sentences $P\hat{c}_j$ on an individual basis, we can try to find a particular id constant $\hat{c}$ such that proving $P\hat{c}$ is tantamount to proving $P\hat{c}_j$ for every $\hat{c}_j$. Then, by substituting $\hat{c}$ for the bullet constant $a^\bullet$ in the modal atoms $[S_i]Pa^\bullet$, we can prove that there is a belief derivation for any referent of $a^\bullet$, just by finding a derivation for the single sentence $P\hat{c}$. Of course, for this scheme to work on a sentence with an arbitrary number of bullet constants, there must be a countably infinite set of id constants that have the requisite property. By placing a minor restriction on the deduction rules $\rho(i)$ or each agent, we can ensure that this is the case.

Consider an id constant $\hat{c}$ and a deduction rule R in which it appears. Suppose that for any other id constant $\hat{c}'$ there is a deduction rule R' that is exactly the same as R, except that every occurrence of $\hat{c}$ is replaced by $\hat{c}'$. A rule R with this property is called *schematic in* $\hat{c}$. Now suppose there is a countably infinite subset of id constants $\hat{c}_j$, such that every rule R of the system $\mathcal{R}$ is schematic in each $\hat{c}_j$. Then we call $\mathcal{R}$ *schematic in* $\hat{c}_j$, and we call the id constants $\hat{c}_j$ *schematic constants* and write them as $\ddot{c}_j$.

The reason schematic constants are important is that any belief derivation of a sentence containing a schematic constant $\ddot{c}$ can be converted into a proof of the same sentence with the $\ddot{c}$ replaced by any other id constant. That this is true should not be surprising, since by the very definition of *schematic*, we are able to replace every step that contains $\ddot{c}$ by a step that contains the new constant.

Now demanding that a rule set be schematic in a countably infinite set of id constants does not seem to be a particularly restrictive condition. We can think of the schematic constants as being a special class of id constants whose interpretation is not known. There can still be id constants, indeed an infinite number of them, that are treated specially by the deduction rules. All we ask is that, in addition, there be schematic constants.

The rules $\mathcal{T}_0$ are schematic in all constants, because they do not treat any of them in a special way. Nonschematic constants occur when we consider deduction rules for languages that are partially interpreted. A good example here would be the term $\ulcorner me \urcorner$ denoting the agent himself. Supposedly, some of an agent's rules would treat sentences involving this constant in a very different manner from other sentences.

The *bullet deletion transform* of a set of formulæ that contain bullet constants is an important concept. The idea is that, in a set of expressions containing bullet constants, we can replace these constants uniformly by schematic constants.

DEFINITION 9.3. *Let* Π *be a set of formulæ from the language* L^{qB}. *Let* $a_1^\bullet \ldots a_n^\bullet$ *be the bullet constants of* Π, *and* $\ddot{c}_1 \ldots \ddot{c}_n$ *be id constants. The* bullet deletion transform *of* Π *is the set of formulæ* $\Pi^\bullet$ *formed by replacing each instance of* $a_i^\bullet$ *in* Π *by its corresponding* $\ddot{c}_i$.

In a slight abuse of notation, we will often write $\Gamma^\bullet, \alpha^\bullet$ to indicate the set $\{\Gamma, \alpha\}^\bullet$.

9.2 Sequent Systems for qB

We turn now to the development of a sequent system axiomatization for qB.

DEFINITION 9.4. *The system* $\mathsf{qB}(L_0, \rho)$ *is given by the following postulates.*

$\Rightarrow$ *The first-order complete rules* $\mathcal{T}_0$.

$$qA: \quad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, \Delta}{\Gamma^\bullet \Vdash_{\rho(i)} \alpha^\bullet}$$

$\Vdash_{\rho(i)}$ *A closed derivation operator for each agent S_i whose rules are schematic in a countably infinite set of id constants $\ddot{c}_j$.*

The definition of qB is the same as that of B, with the exception of the attachment rule qA and the restriction on ρ. It is impossible to have Γ and α appear unmodified in belief derivation, because they will not be legal sentences if they contain bullet constants, *e.g.*, $[S_i]Pa^\bullet$ would yield formula $Pa^\bullet$, which is not a sentence of L^{qB}. Because the difference between qA and A affects only quantified-in statements (the bullet deletion transform of a sentence without bullet constants is itself), it is trivial to show that $\mathsf{B}(L_0, \rho)$ and $\mathsf{qB}(L_0, \rho)$ have exactly the same theorems from the language L^{B}.

9.2.1 *Some Theorems of* qB

THEOREM 9.1. *Suppose $Pa \Vdash_{\rho(i)} \exists x.Px$ (existential generalization) holds for every constant a in agent S_i's belief subsystem. Then*

$$\vdash_{\mathsf{qB}} \exists x.\, [S_i]Px \Rightarrow [S_i]\exists x.Px \quad .$$

Proof.

$$\begin{array}{ll} E_2 & \dfrac{\exists x.[S_i]Px \Rightarrow [S_i]\exists x.Px}{I_i a, [S_i]Pa^\bullet \Rightarrow [S_i]\exists x.Px} \\ qA & \dfrac{}{P\ddot{c} \Vdash_{\rho(i)} \exists x.Px} \end{array}$$

which closes by assumption.∎

THEOREM 9.2.

$$\nvdash_{\mathsf{qB}} [S_i]\exists x.Px \Rightarrow \exists x.\, [S_i]Px$$

Proof.

$$C_2\;\dfrac{E_2\;\dfrac{[S_i]\exists x.Px \Rightarrow \exists x.\,[S_i]Px}{[S_i]\exists x.Px \Rightarrow \exists x.\,[S_i]Px,\, I_ia \wedge [S_i]Pa^\bullet}}{[S_i]\exists x.Px \Rightarrow \exists x.\,[S_i]Px,\, [S_i]Pa^\bullet \qquad [S_i]\exists x.Px \Rightarrow \exists x.\,[S_i]Px,\, I_ia}$$

The right-hand branch will never close, because each time the existential rule E_2 is used, a new conjunct involving I_ia is created.∎

THEOREM 9.3.

$$\nvdash_{\mathsf{qB}} [S_i]P\hat{c} \Rightarrow \exists x.[S_i]Px$$

Proof.

$$C_2\;\dfrac{E_2\;\dfrac{[S_i]P\hat{c} \Rightarrow \exists x.\,[S_i]Px}{[S_i]P\hat{c} \Rightarrow \exists x.\,[S_i]Px,\, I_ia \wedge [S_i]Pa^\bullet}}{[S_i]P\hat{c} \Rightarrow \exists x.\,[S_i]Px,\, [S_i]Pa^\bullet \qquad [S_i]P\hat{c} \Rightarrow \exists x.\,[S_i]Px,\, I_ia}$$

The right-hand branch cannot close.∎

These three theorems show that quantifying into existential contexts has the required properties; that is, if for some x an agent has the belief Px, it is an easy inference for him to believe $\exists x.Px$ (Theorem 9.1); but the converse is not true (Theorem 9.2). More strongly, by Theorem 9.3 even if an agent has the belief $P\hat{c}$ for some id constant $\hat{c}$, from the point of view of an outside observer he still may not have the belief Px for some individual x, because there may be no naming map from any real individual to the id constant $\hat{c}$.

We now show that neither the Barcan formula $\forall x.[S_i]Px \supset [S_i]\forall x.Px$ nor its coverse hold in **q**B.

THEOREM 9.4.

$$\nvdash_{\mathsf{qB}} \forall x.[S_i]Px \Rightarrow [S_i]\forall x.Px$$

Proof.

$$\begin{array}{ll} & \forall x.[S_i]Px \Rightarrow [S_i]\forall x.Px \\ U_1 & \overline{I_i a \wedge [S_i]Pa^\bullet \Rightarrow [S_i]\forall x.Px} \\ C_1 & \overline{I_i a, [S_i]Pa^\bullet \Rightarrow [S_i]\forall x.Px} \\ qA & \overline{P\ddot{c} \nvdash_{\rho(i)} \forall x.Px} \end{array}$$

This branch is open for sound belief deduction.▮

THEOREM 9.5.

$$\nvdash_{\mathsf{qB}} [S_i]\forall x.Px \Rightarrow \forall x.[S_i]Px$$

Proof.

$$\begin{array}{ll} & [S_i]\forall x.Px \Rightarrow \forall x.[S_i]Px \\ U_2 & \overline{[S_i]\forall x.Px \Rightarrow [S_i]Pa^\bullet \wedge I_i a} \\ C_2 & \overline{[S_i]\forall x.Px \Rightarrow [S_i]Pa^\bullet \qquad [S_i]\forall x.Px \Rightarrow I_i a} \end{array}$$

The right-hand branch cannot close.▮

Both these theorems are to be expected on the basis of the informal semantics of quantified-in sentences. The Barcan formula fails to hold because there may be some id constants that are not the names of any real individual, even if it is asserted for every individual x that Px is a belief. The converse fails to hold because there may be some individual x that does not have an id name, and so Px will not be a belief even if all sentences of the form Pa are beliefs. Note the importance of introducing the I-predicate in substitution. It is the presence of this predicate that causes the right-hand branch in Theorems 9.2, 9.3, and 9.5 to remain open.

9.3 Model Theory for qB

We define $\mathsf{qB}(L_0, \rho)$-models in a manner similar to $\mathsf{B}(L_0, \rho)$-models, namely, as a first-order valuation for L_0 and a set of deduction structures from the classes $\mathbf{D}(L, \rho(\uplus))$. However, there are some changes. There is an additional element η_i for each agent S_i, a naming function from individuals to id constants of L^{qB}. Note that each η_i is a *total* function from individuals to names. The partial nature of the name map is preserved by the predicates I_i, which have a special interpretation in the model: for each agent S_i, they pick out the individuals that may validly be referred to by the naming function.

DEFINITION 9.5. *A* $\mathsf{qB}(L_0, \rho)$-model *is a tuple* $\langle v_0, \varphi, \boldsymbol{\eta}, \mathsf{U}, D\rangle$, *where* v_0 *is an atomic valuation of* E^{U}, φ *is a mapping from constants to elements of* U, $\boldsymbol{\eta}$ *is a sequence of total functions from elements of* U *to id constants of* L^{qB}, *and* D *is a sequence consisting of one member from each of the classes* $\mathbf{D}(L, \rho(i))$, $i > 0$. *The rules* $\rho(i)$ *must be schematic under the countably infinite set of constants* $\ddot{c}_j$.

We wish to extend the definition of the valuation function V to sentences of the language L^{qB}. V will have the same behavior on sentences of L^{B} as it did previously, since every $\mathsf{qB}(L_0, \rho)$-model is also a $\mathsf{B}(L_0, \rho)$-model. Recall from Definition 4.2 of V that the part dealing with quantification subsitutes elements of U for variables. We need to define the substitution of U-elements into the context of modal operators, by appending the following rule to Definition 9.2:

4. $([S_i]\alpha)^x_k = [S_i]\alpha^x_k$

We thus allow elements of U to appear in the argument of belief operators. The semantics of such elements is that they stand for the id constant given by the naming map.

A model m defines an interpretation of a modal atom's argument as a sentence of L^{qB}. For example, consider the atom $[S_i]P(k, a^\bullet)$; the subexpression $\ulcorner P(k, a^\bullet)\urcorner$ refers to the sentence $\ulcorner P(\hat{c}_k, \hat{c}_a)\urcorner$, where $\hat{c}_k = \eta_i(k)$, and $\hat{c}_a = \eta_i(\varphi(a))$. Note that the id constant assigned to bullet terms depends on the interpretation φ of unspecific constants, as well as the naming function η_i. We write $[\![\phi]\!]^i_m$ to indicate the interpretation of the argument to a modal operator $[S_i]$ in the model m. We abbreviate the set $\{[\![\gamma_1]\!]^i_m, [\![\gamma_2]\!]^i_m, \ldots\}$ as $[\![\Gamma]\!]^i_m$.

We now give the valuation function V for the language L^{qB}.

DEFINITION 9.6. *Let $m = \langle v_0, \varphi, \eta, \mathsf{U}, D\rangle$ be a $\mathsf{qB}(L_0, \rho)$-model, and s a sentence of L^{qB}. The valuation function $V(s,m)$ is defined by the following rules.*

1. *$V(s,m)$ is a first-order valuation that agrees with v_0 and φ when s is nonmodal.*
2. *$V([S_i]p, m) = \mathsf{t}$ iff $[\![p]\!]^i_m \in \mathrm{bel}(d_i)$, where $d_i \in D$, and $V(I_i k, m) = \mathsf{t}$ for every element k of p.*

Note that this definition of V does indeed subsume the older one, for $[\![p]\!]^i_m = p$ if $[S_i]p$ contains no bullet constants or elements of U. Also, because the naming map is partial, there may be no image of an element k of U; if this is the case, then the value of $[S_i]p$ is f, because $I_i k$ will be false.

A $\mathsf{qB}(L_0, \rho)$-*interpretation* of the sentences of L^{qB} is an assignment of truth-values produced by the valuation function with respect to some model. A sentence of L^{qB} is $\mathsf{qB}(L_0, \rho)$-*satisfiable* just in case it is true in some $\mathsf{qB}(L_0, \rho)$-interpretation, and $\mathsf{qB}(L_0, \rho)$-*valid* exactly when it is true in all $\mathsf{qB}(L_0, \rho)$-interpretations. We will write $m \models s$ if $V(s,m) = \mathsf{t}$, and $\mathsf{qB}(L_0, \rho) \models s$ if s is $\mathsf{qB}(L_0, \rho)$-valid.

THEOREM 9.6. *If s is a sentence of L^{B} based on L_0, and ρ is schematic, then*

$$\mathsf{qB}(L_0, \rho) \models s \;\leftrightarrow\; \mathsf{B}(L_0, \rho) \models s\ .$$

Proof. Every $\mathsf{qB}(L_0, \rho)$-model contains a $\mathsf{B}(L_0, \rho)$-model that has the same behavior on sentences of L^{B}. On the other hand, given a $\mathsf{B}(L_0, \rho)$-model in which ρ is schematic, adding any naming function ensemble η constructs a $\mathsf{qB}(L_0, \rho)$-model that gives the same valuation on sentences of L^{B}, because there are no bullet constants or quantified-in variables.∎

We have defined substitution in this complicated manner so that it obeys the following important property. Suppose ψ is a formula with a single free variable x, such that m satisfies ψ when free x is everywhere replaced by the element k of m:

$m \models \psi^x_k$. Let a be a constant whose interpretation in m is the element k, so that $\varphi(a) = k$. We would expect m to also satisfy the formula ψ when a was substituted for free x: $m \models \psi^x_a$. This is indeed the case, as we now prove.

THEOREM 9.7. *Let ψ be a formula whose only free variable is x, and let $m \models \psi^x_k$, where k is an element of m. If a is a constant such that $\varphi(a) = k$, then $m \models \psi^x_a$.*

Proof. Let ϕ be an atom appearing in ψ, and let ϕ' be this atom under the substitution ψ^x_k, ϕ'' under ψ^x_a. We will show that every ground instance of ϕ' has the same truthvalue as that of ϕ''.

If ϕ is an ordinary atom $P(\mathbf{y}, x)$, then, in any interpretation for which $\varphi(a) = k$, we have $v_0(P(\mathbf{k}, a)) = v_0(P(\mathbf{k}, k))$, and hence $v_0(\phi'^{\mathbf{y}}_{\mathbf{k}}) = v_0(\phi''^{\mathbf{y}}_{\mathbf{k}})$.

If ϕ is a modal atom $[S_i]\alpha$ with free variables $\mathbf{y}$ and x, then $\phi' = [S_i]\alpha^x_k$ and $\phi'' = I_i a \wedge [S_i]\alpha^x_{a^\bullet}$. Suppose $I_i k$ is true in m. Then $I_i a$ is also true, and the valuation of $\phi''^{\mathbf{y}}_{\mathbf{k}}$ is equal to the valuation of $[S_i]\alpha^{x,\mathbf{y}}_{a^\bullet,\mathbf{k}}$. We know that $\eta_i(k) = \eta_i(\varphi(a))$, so that $[\![\alpha^{x,\mathbf{y}}_{k,\mathbf{k}}]\!]^i_m = [\![\alpha^{x,\mathbf{y}}_{a^\bullet,\mathbf{k}}]\!]^i_m$, and hence $\phi'^{\mathbf{y}}_{\mathbf{k}}$ has the same value as $\phi''^{\mathbf{y}}_{\mathbf{k}}$. On the other hand, suppose $I_i k$ is false in m; then both $\phi'^{\mathbf{y}}_{\mathbf{k}}$ and $\phi''^{\mathbf{y}}_{\mathbf{k}}$ are false.∎

9.3.1 Soundness and Completeness

We now prove the soundness and completeness of **qB** with respect to its models, in much the same way that we proved these properties for **B** in Chapter 4. Complications arise because of quantified-in variables and bullet constants.

LEMMA 9.8.

$$\Gamma^\bullet \Vdash_{\rho(i)} \alpha^\bullet \quad \rightarrow \quad \mathsf{qB} \models [S_i]\Gamma \Rightarrow [S_i]\alpha$$

Proof. Suppose to the contrary that $[S_i]\Gamma \Rightarrow [S_i]\alpha$ is not valid. Then there is a model m that falsifies this sequent. If $d_i \in \mathbf{D}$ is the deduction structure of m for agent S_i, then we must have $[\![\alpha]\!]^i_m \notin \text{bel}(d_i)$ and $[\![\Gamma]\!]^i_m \subseteq \text{bel}(d_i)$. But since there is a proof $\Gamma^\bullet \Vdash_{\rho(i)} \alpha^\bullet$ with $\rho(i)$ schematic, there is also a proof of $[\![\Gamma]\!]^i_m \Vdash_{\rho(i)} [\![\alpha]\!]^i_m$. Thus by the closure property of $\Vdash$, $[\![\alpha]\!]^{i}_{m}$ must be included in bel(d_i), a contradiction.∎

The fact that the rules $\rho(i)$ were schematic was an essential part of the proof of this lemma. If they were not, the derivation $\Gamma^{\bullet} \Vdash_{\rho(i)} \alpha^{\bullet}$ would be no guarantee of $[S_i]\Gamma \Rightarrow [S_i]\alpha$ being valid, since the belief derivation might have depended on special id constants in the bullet deletion transform.

THEOREM 9.9. (Soundness of qB)

$$\vdash_{\mathsf{qB}} \Gamma \Rightarrow \Delta \quad \rightarrow \quad \mathsf{qB} \models \Gamma \Rightarrow \Delta$$

Proof. Consider a closed tableau for some theorem of qB. We would like to show that the root of the tree is valid. We do this by showing that the parent is valid whenever a set of daughters is valid, *i.e.*, the deduction rules preserve validity. Then if the axioms are valid, the root node (and indeed every node in the tableau) must be valid.

For the first-order rules $\mathcal{T}_0$, we know already that the propositional rules preserve validity. The quantificational rules can involve substitution into modal contexts; we use Theorem 9.7 to show that they are sound. Consider the rule U_1, and assume that the top sequent is not valid, so that $\{\Gamma, \forall x.\phi, \neg\Delta\}$ is satisfied by some qB-model m. But then by Theorem 9.7, $m \models \phi^x_a$ because $m \models \phi^x_k$ for every k in U, and so the set $\{\Gamma, \forall x.\phi, \phi^x_a, \neg\Delta\}$ is also satisfied by m. Hence the bottom sequent is not valid.

For the rule U_2, again assume that the top sequent is not valid, so that $\{\Gamma, \neg\forall x.\phi, \neg\Delta\}$ is satisfied by some m. Because $\neg\forall x.\phi$ is equivalent to $\exists x.\neg\phi$, there is some constant k such that $m \models \neg\phi^x_k$. Now the constant a does not appear in any of the sentences of the set, so we can construct a model m' that is exactly the same as m, but assign the interpretation $\varphi(a) = k$. By Theorem 9.7, we must have $m' \models \neg\phi^x_a$; thus the extended set $\{\Gamma, \neg\forall x.\phi, \neg\phi^x_a, \neg\Delta\}$ is satisfied by m', and the bottom sequent is not valid.

The proof for the existential rules is similar.

For the attachment rule qA, the top sequent must be valid by Lemma 9.8.∎

Remarks. We have proven the soundness of **qB** relative to a closed, schematic derivation operator for each agent. Note that the condition of soundness on an agent's rules was not used in the proof of this theorem.

To prove completeness, we first prove a lemma that is the converse of Lemma 9.8.

LEMMA 9.10. (q-Attachment) *Let* $Z =_{\rm df} \{[S_i]\Gamma', \neg[S_i]\Delta'\}$ *be a (perhaps denumerably infinite) set that is* qB*-unsatisfiable. Then for some* $\delta \in \Delta'$ *and finite* $\Gamma \subseteq \Gamma'$, $\Gamma^\bullet \Vdash_{\rho(i)} \delta^\bullet$.

Proof. Assume that $\Gamma^\bullet \nVdash_{\rho(i)} \delta^\bullet$ for all sentences $\delta \in \Delta'$ and all finite subsets $\Gamma \subseteq \Gamma$. Under this assumption we can construct a model m that satisfies Z as follows. Choose an arbitrary v_0 and a denumerably infinite universe U, and define φ so that each constant refers to a different individual of U. Suppose that the bullet deletion transform converts $a_j^\bullet$ to the schematic constant $\ddot{c}_j$ for agent S_i; choose η_i such that $\eta_i(\varphi(a_j)) = \ddot{c}_j$. With these choices, we have $[\![\{\Gamma', \Delta'\}]\!]_m^i = \{\Gamma'^\bullet, \Delta'^\bullet\}$. Now construct a deduction structure $d_i =_{\rm df} \langle [\![\Gamma']\!]_m^i, \rho(i)\rangle$; this has the property that no member of $[\![\Delta']\!]_m^i$ is in bel(d_i), because by assumption there is no such belief derivation. Hence, for the qB-model $m =_{\rm df} \langle v_0, \varphi, \mathsf{U}, \boldsymbol{\eta}, \{\ldots d_i \ldots\}\rangle$ we have $m \models [S_i]\gamma$ for each $\gamma \in \Gamma'$, and $m \not\models [S_i]\delta$ for each $\delta \in \Delta'$. This is a contradiction, since it was assumed that no such model existed.∎

COROLLARY 9.11. *For some* $\delta \in \Delta$,

$$\mathsf{qB} \models [S_i]\Gamma \Rightarrow [S_i]\Delta \quad \rightarrow \quad \Gamma^\bullet \Vdash_{\rho(i)} \delta^\bullet$$

The attachment lemma is the main step in proving completeness, since it relates the validity of sequents involving atomic belief sentences to their provability. It is called an *attachment* lemma in analogy to techniques of semantic attachment, in which the validity of a sentence is shown by attaching to the intended meaning of the sentence in a partial model, and computing a truthvalue (see, *e.g.*, Weyhrauch [66]. In the attachment lemma, the validity of a sentence involving modal atoms is determined by computation in its intended model, namely, deduction in a belief subsystem.

This version of the attachment lemma is essentially the same as Lemma 4.4 for the logic B, except that belief derivation uses the bullet deletion transform to eliminate bullet constants in favor of schematic constants.

THEOREM 9.12. (Completeness of qB) *Let* Γ *and* Δ *be finite subsets of* Γ' *and* Δ', *respectively. Then*

$$\mathsf{qB} \models \Gamma' \Rightarrow \Delta' \quad \rightarrow \quad \vdash_{\mathsf{qB}} \Gamma \Rightarrow \Delta \, .$$

Proof. The proof is similar to that of Theorem 4.6. We define the set W as the Hintikka set constructed from an open branch of the tableau. W contains a (perhaps infinite) set Z of modal atoms $[S_i]\Gamma'_i$ and negations of modal atoms $\neg[S_i]\Delta'_i$ for each agent S_i. We will show that each such set is qB-satisfiable.

Because the branch is open, there is no finite subset $\Gamma_i \subseteq \Gamma'_i$ such that $\Gamma^\bullet_i \not\vdash_{\rho(i)} \delta^\bullet$ Hence, by the contrapositive of the Q-Attachment Lemma, the set $\{[S_i]\Gamma'_i, \neg[S_i]\Delta'_i\}$ is satisfiable.∎

The compactness of qB follows immediately from the completeness theorem.

COROLLARY 9.13. (Compactness of qB) *If a set of sentences of* L^{qB} *is unsatisfiable, it has a finite unsatisfiable subset.*

9.4 Correspondence Property for qB

We now interpret L^{qB} in a Kripkean possible-world model, and prove that saturated qB, restricted in an appropriate fashion, is sound and complete with respect to this model. We restrict our attention here to the nonintrospective form of qB, in which the belief derivation operator in qA is replaced with a sequent sign, as in BK. The obvious modifications for introspective behavior can be found by analogy with the systems BS4 and BS5 of Chapter 7.

9.4.1 *Quantificational Kripke Models for Belief*

We now define quantificational possible-world models (from Kripke [36]). A *model structure* is a triple $\langle w_0, \mathcal{W}, R\rangle$, where $\mathcal{W}$ is a set of worlds, w_0 is an element of $\mathcal{W}$ (the real world), and R is a binary relation on worlds. Kripke takes R to be reflexive, but this makes every belief true (the schema $[S_i]p \supset p$ is valid), and so is not assumed here. A *model* is a model structure together with valuation functions defined as follows. Associated with each world w is a universe U_w, an atomic valuation v_w of predicates, and a denotation map φ_w for constants. The truth of any ordinary ground atom in w can be determined with v_w and φ_w. Any ordinary ground modal atom $[S_i]p$ is true in w just in case p is true in every w' such that wRw'. Quantifiers are assumed to range over only those individuals present in a world, so that $\exists x.Q(x)$ is true in a world w just in case $Q(x)$ is true of some individual in U_w.

Kripke models for quantified modal logic must be modified slightly to correspond to our intuitions about belief. Note that $P^n(k_1, \ldots k_n)$ is assigned a truth-value in w even when some of the k_j are not in U_w. This isn't quite what we would like as an interpretation for belief. Consider the sentence $\exists x.[S_i]Px$; it will be true in a quantificational Kripke model that has Pk true in each compatible possible world w, even if k is not a member of the universe U_w of this world. It seems likely that, if we assert that there is an individual such that S_i believes P of the individual, we mean also that S_i believes the individual to exist. This, in fact, corresponds to the interpretation that we gave in Definition 9.6, where we assume that a belief atom $[S_i]Pk$ is false unless k is an individual that S_i is aware of, in the sense that k is in the naming map (I_ik is true).

The problem of interpreting Pk in a world w for which k does not exist can be avoided if we always assume that either

$$(KR_1) \qquad \mathsf{U}_w \subseteq \mathsf{U}_{w'}$$

or

$$(KR_2) \qquad \mathsf{U}_w = \mathsf{U}_{w'}$$

for all worlds w and w' such that wRw'. The condition KR_1 assures us that every individual quantified over in a world will be present in all compatible worlds, so that the problem of assigning truthvalues to predicates involving individuals outside a world's universe does not arise. If the sentence $\exists x.[S_i]Px$ is true in w, then there is an individual k of U_w such that Pk is true in every compatible world w'; by condition KR_1, k will always be in U'_w. Philosophically, KR_1 commits us to the view that every agent is cognizant of, or can have beliefs about, all real individuals. Formally, it makes the converse Barcan formula $[S_i]\forall x.A(x) \supset \forall x.[S_i]A(x)$ valid, because if $A(k)$ is true of every individual in $\mathsf{U}_{w'}$, it is also true of every individual in U_w.

The second condition, KR_2, is strictly stronger than the first, and might be called the assumption of a common universe. There are two ways to interpret this condition. The first is to take the view that every agent is cognizant of exactly the real individuals. This is somewhat restrictive, because it assumes that agents will not have any beliefs about possible, but nonexistent, individuals — such as Santa Clause or griffins. An alternate interpretation is advanced by Moore [51]. He assumes that the common universe includes all possible individuals as well as the real ones. That is, a quantified variable ranges over any individuals that any agent believes to exist, even if they are not actual individuals. Essentially this proposal amounts to identifying the universe of every possible world as the set of all real and possible individuals. Of course, the interpretation of ordinary quantified expressions is now somewhat different: $\exists x.Qx$ states that some real or possible individual has the property Q, rather than being restricted to real individuals. A distinguished existence predicate is necessary if we wish to differentiate the real from the possible individuals in a world.

The common universe condition makes both the Barcan formula $\forall x.[S_i]A(x) \supset [S_i]\forall x.A(x)$ and its converse valid in quantificational Kripke models. To see this for the Barcan formula, suppose we assume that $\forall x.[S_i]A(x)$ is true in w, so that in every compatible world w', Pk is true for all individuals k in U_w. Because $\mathsf{U}_w = \mathsf{U}_{w'}$, $\forall x.A(x)$ must be true in w' as well.

There are qB-analogs to each of the conditions KR_1 and KR_2. If we take the interpretation of $I_i k$ to mean that k is an individual that S_i can have beliefs about, then KR_1 would render $I_i k$ valid for every individual k. With this restriction, the converse of the Barcan formula $([S_i]\forall x.A(x) \Rightarrow \forall x.[S_i]A(x))$ becomes a theorem of saturated qB. To see this, note that the right-hand branch in the tableau of Theorem 9.5 now closes, because $I_i a$ is true.

A further addition to qB is needed to accommodate the common universe condition. Consider the open tableau that we used to show the invalidity of the Barcan formula:

$$\begin{array}{ll}
 & \forall x.[S_i]Px \Rightarrow [S_i]\forall x.Px \\ \cline{2-2}
U_1 & I_i a \wedge [S_i]Pa^{\bullet} \Rightarrow [S_i]\forall x.Px \\ \cline{2-2}
C_1 & I_i a, [S_i]Pa^{\bullet} \Rightarrow [S_i]\forall x.Px \\ \cline{2-2}
qA & P\ddot{c} \nVdash_{\rho(i)} \forall x.A(x) \\
 & \times
\end{array}$$

This tableau fails to close because it is impossible to soundly infer that $\forall x.Px$ is true when it is only known to be true of the individual $\ddot{c}$. However, given the assumption of a common universe, we might argue as follows. Although $\ddot{c}$ refers to a single individual, it is an arbitrarily chosen individual. When its antecedent a was chosen as the instantiation constant in the rule U_1, it was a new constant, and hence could stand for any individual in the universe U_{w_0}. Because $\mathsf{U}_{w_0} = \mathsf{U}_w$ for any compatible world w, $\ddot{c}$ represents an arbitrarily chosen individual for the agent S_i. If $P\ddot{c}$ is true when $\ddot{c}$ is arbitrarily chosen, it must be true for all individuals.

In the following subsections we prove the first of these correspondences, namely, that the saturated, nonintrospective version of qB with all I_i predicates identically true has the same theorems as the axiomatization of quantificational Kripke models with condition KR_1.

9.4.2 *Interpreting* qB *in Possible Worlds*

The first part of the correspondence proof is to show that the system qB, when suitably interpreted, is sound with respect to the possible-world model. This is a different technique from the proof for the propositional case in Section 8.1.3, where we simply showed that every proof in saturated B had a corresponding proof in a normal modal logic. Here, the presence of bullet terms generated in the course of a tableau proof makes this technique unsuitable, because these terms are not present in axiomatizations of Kripke models. Also, interpreting L^{qB} with respect to Kripke models is a useful exercise because it gives insight into the nature of the language.

To this end, we first note the following restrictions on qB, as discussed above. This system is called qBK_s.

1. The I_i predicates are identically true, and can be eliminated from the definition of substitution into modal contexts.
2. Belief derivation is treated as proof in a sequent system, and all agent's rules $\tau(\nu)$ are assumed to be the same as the outside observer's. The index on sequent signs can be eliminated, and the attachment rule qA becomes:

$$qA_K : \quad \frac{\Sigma, [S_i]\Gamma \Rightarrow [S_i]\alpha, \Delta}{\Gamma^\bullet \Rightarrow \alpha^\bullet}$$

To the interpretation rules for quantificational Kripke models given in the last section, we must add a rule for bullet constants. Let $[S_i]p$ be a closed belief atom which contains a bullet constant $a^\bullet$, and let p' be p with $a^\bullet$ everywhere replaced by

$\varphi_w(a)$. The belief atom is true in w if and only if the sentence p' is true in every w' such that wRw'. From this and the previous rules, the truth of every sentence in L^{qB} with respect to a world can be determined; we write $w \models s$ if s is true in w. A sentence s is true in a model *iff* $w_0 \models s$; it is valid *iff* it is true in every model. We will use one fact about Kripke models: every valid sentence is true in every world of every model (see Huges and Cresswell [23], page 351).

We now prove the soundness theorem.

THEOREM 9.14. *The rules of* qBK_s *are sound with respect to the modified quantificational Kripke models.*

Proof. The propositional rules of $\mathcal{T}_0$ are obviously sound. For the quantificational rules, consider first U_1, and assume that the top sequent is not valid, so that $\{\Gamma, \forall x.\psi, \neg\Delta\}$ is satisfied by some Kripke model m. We wish to show that the sentence ψ^x_a is true in m. We must show that for every atom ϕ in ψ, $w_0 \models \phi^x_a$ *iff* $w_0 \models \phi^x_k$, where $\varphi_{w_0} = a$. It is obvious for ordinary atoms. For a modal atom $\phi =_{\mathrm{df}} [S_i]p$, we have $\phi^x_a = [S_i]p^x_{a^\bullet}$. The valuation of $a^\bullet$ is the element k such that $\varphi_{w_0}(a) = k$; hence $[S_i]p^x_{a^\bullet}$ has the same value in w_0 as $[S_i]p^x_k$.

For the rule U_2, again assume that the top sequent is not valid, so that $\{\Gamma, \neg\forall x.\psi, \neg\Delta\}$ is satisfied by some Kripke model m. Because $\neg\forall x.\psi$ is equivalent to $\exists x.\neg\psi$, there is some constant k such that $w_0 \models \neg\psi^x_k$. Now the constant a does not appear in any of the sentences of the set, so we can construct a model m' that is exactly the same as m, but assign the interpretation $\varphi_{w_0}(a) = k$. By the arguments of the previous paragraph, $w_0' \models \neg\psi^x_a$.

The proof for the existential rules is similar.

For the attachment rule, assume that the lower sequent $\Gamma^\bullet \Rightarrow \alpha^\bullet$ is valid. Let Γ' and α' be the sentences obtained by uniformly replacing all schematic constants $\ddot{c}_j$ introduced by the bullet deletion operation with different individuals k_j. The sequent $\Gamma' \Rightarrow \alpha'$ must be valid; if it were not, there would be a model m whose w_0 falsified this sequent, and we could then construct a model m' with $\varphi_{w_0'}(\ddot{c}_j) = k_j$ that falsified $\Gamma^\bullet \Rightarrow \alpha^\bullet$.

Let m be an arbitrary model with actual world w_0, and let $a^\bullet_j$ be the bullet constants of $\Gamma^\bullet$ and $\alpha^\bullet$. Form Γ' and α' from $\Gamma^\bullet$ and $\alpha^\bullet$ by the substitution $k_j = \varphi_{w_0}(a_j)$. Because $\Gamma' \Rightarrow \alpha'$ is valid, it must be true in every world w compatible with w_0. By the valuation rules for bullet constants, $[S_i]\Gamma \Rightarrow [S_i]\alpha$ must be valid in w_0.∎

9.4.3 *Completeness of* qBK$_s$

We prove the completeness of qBK$_s$ by showing that it has the same theorems (over L^{qB} without bullet terms) as the axiomatization of quantificational Kripke models with the KR_1 condition. The *closure* of a formula A is any sentence obtained from A by prefixing universal quantifiers and modal operators, in any order. The axioms are the closure of the following schemata (from Kripke [36]):

KR_0 :	All tautologies.
KR_1 :	$[S_i]\forall x.A(x) \supset \forall x.[S_i]A(x)$
KR_3 :	$[S_i](A \supset B) \supset ([S_i]A \supset [S_i]B)$
KR_4 :	$A \supset \forall x.A$, where x is not free in A.
KR_5 :	$\forall x.(A \supset B) \supset (\forall x.A \supset \forall x.B)$
KR_6 :	$\forall y.(\forall x.A(x) \supset A(y))$

The sole rule of inference is *modus ponens*; necessitation can be obtained as a derived rule. We have modified Kripke's original axioms by deleting the schema $[S_i]A \supset A$, and introducing the converse Barcan formula KR_1.

In showing that the closures of the schemata are theorems in qBK$_s$, the following fact will be useful.

LEMMA 9.15. *Let T be a sentence of L^{qB} that has the sentence p as one of its subexpressions, such that a proof exists for T no matter what p is. Let $A(x)$ be a formula with the free variable x, which does not occur in T. Let T' be T with p everywhere replaced by $A(x)$. Then $[S_i]T$, $\forall x.T'$, and $\forall x.[S_i]T'$ are all theorems of* qBK$_s$.

Proof. For $[S_i]T$, the attachment rule yields the sequent $\Rightarrow T$, which closes by assumption.

For $\forall x.T'$, one application of U_2 yields the sequent $\Rightarrow T'^{x}_{a}$. Because there is a proof of T for arbitrary p, there is one for p replaced by $A(a)$.

Note that this depends on the fact that qBK_s does not introduce the predicate I_i in instantiations.

For $\forall x.[S_i]T'$, one application of U_2 yields the sequent $\Rightarrow [S_i]T'^{x}_{a}$, and the attachment rule applies to give $\Rightarrow T'^{x}_{a}$. Again, there is a proof of this sequent.▮

We now prove the completeness of qBK_s relative to quantificational Kripke models for belief.

THEOREM 9.16. *Every theorem of the Kripke KR system is a theorem of* qBK_s.

Proof. From the proof of Theorem 8.6, we know that the theorems of qBK_s are closed under *modus ponens*, and include the tautologies.

We show that each of the *KR* schemata is a theorem of qBK_s when the schema variables A and B are taken to be sentences. By Lemma 9.15, substituting formulas with free variables and taking the closure also yields theorems of qBK_s.

For KR_1:

$$U_1\;\dfrac{qA_K\;\dfrac{U_2\;\dfrac{[S_i]\forall x.Ax \Rightarrow \forall x.[S_i]Ax}{[S_i]\forall x.Ax \Rightarrow [S_i]Aa^{\bullet}}}{\forall x.Ax \Rightarrow A\ddot{c}}}{\dfrac{A\ddot{c} \Rightarrow A\ddot{c}}{\times}}$$

We leave the rest of the schemata as exercises for the reader.▮

10. Proof Methods: Davis-Putnam Generalized for B$^+$

We now turn to the investigation of practical automatic theorem-proving methods for the various belief logics that have been defined. These fall into two classes, depending on the expressiveness of the language: decision procedures for the propositional case, and resolution-based systems for the quantified-in case. In this chapter we confine ourselves to propositional logics, in which there is no quantification. The method of Davis and Putnam [9] is an efficient decision procedure for propositional logic, but is not a decision procedure for the modal logic B because the modal atoms have a specific interpretation. We first generalize the procedure to take into account predicates with specific interpretations. The generalization is far from trivial, and an important result in its own right.

Once we have developed the generalized Davis-Putnam method, we can apply it to the case of the logic B$^+$ in which the belief derivation operation $\Vdash_{\rho(i)}$ is decidable for each agent S_i. Because of the correspondence property, the method will also work for propositional modal logics of belief based on possible-world models. The method has been implemented and is efficient, especially in the solution to belief puzzles like the Wise Man Puzzle. The appendix contains a trace of the computer proof of the so-called hard form of the puzzle (in which the ignorance of the first two wise men is shown) using the saturated form of the logic B$^+$. This proof may be compared with the block tableau proof in Section 6.3.

Given the importance and prevalence of current AI research in logics for knowledge and belief, it is surprising that more attention has not been given to developing efficient decision procedures for the propositional case. Decision procedures for propositional T, $S4$, and $S5$ using tableau methods were developed by Kripke [36], and, more recently, Sato [61] has done the same for extensions to these logics that correspond to B^+. However, these procedures are not computationally efficient, mostly because of the combinatorics introduced by disjunction. Any tableau system that makes indiscriminate use of rules such as C_2 and D_2 of $\mathcal{T}_0$, in which the tableau is split into two parts, suffers from this computational problem. If there are n such splits, there will be 2^n branches of the tableau, and much redundant work will be done to show that each branch closes.

The Davis-Putnam method achieves its efficiency by delaying splits as much as possible, and reducing to a minimum the number of sentences that are introduced on each side of the split. When adapted to work for propositional modal languages, the method retains these advantages. As far as the author knows, this is the first computationally realistic decision procedure developed for a propositional modal logic of belief.

10.1 A Generalized Davis-Putnam Method

The method of Davis and Putnam tests for the unsatisfiability of a finite set of clauses of the propositional calculus. We will first define the notion of clause, literal, and related concepts, then exhibit a generalized form of Davis-Putnam procedure that can be used for languages whose predicates have particular interpretations; we will use propositional modal languages as an example of this type. The general procedure can be particularized to the unquantified form of B^+ (boolean combinations of ground atoms) whenever there is an effective means of deciding the belief derivation operation $\Gamma \not\Vdash p$. Finally, we exhibit a complete decision procedure for the saturated logic B_s^+.

10.1.1 Clause Form for Propositional Modal Languages

Clause form for a modal language without quantification is similar to that of the propositional calculus, with all ground atoms treated as distinct propositional variables.

> DEFINITION 10.1. *A* literal *is either an atom (modal or ordinary) or its negation. The* complement *of a positive literal p is $\neg p$, and the* complement *of the negative literal $\neg p$ is p. The complement of a literal L is symbolized by $\sim L$.*
>
> *A clause is a finite set of literals interpreted as a disjunction. An empty clause is false in every interpretation. A sentence is in* conjunctive normal form *(CNF) if it is a conjunction of clauses.*

At times we write clauses with disjunctions instead of set brackets, as in $L_1 \vee L_2 \ldots$. We use capital A to stand for a set of literals; the notation $C = L, A$ or $C = L \vee A$ indicates a clause C consisting of the set $\{L\} \cup A$. The null clause is written as ■.

Every set of sentences of a propositional logic is equivalent to a set of clauses. The *clause form* of a set of sentences is such an equivalent set of clauses.

> DEFINITION 10.2. *The* clause form *of a set of unquantified sentences B is the set of clauses obtained by converting each sentence of B to its conjunctive normal form (for example, using the procedure in Robinson [59], pages 150–152).*

Consider a typical sentence of unquantified B^+ that we wish to convert to clause form.

$$(Pa \vee (\neg Q \wedge [S_1, t_1]\exists x.Px)) \vee \langle S_2 : \exists x.Px\rangle Q \tag{10.1}$$

Note that quantifiers can occur within the scope of modal atoms, but the basic structure of (10.1) is a boolean combination of ground atoms. In any interpretation of B^+, each atom has a truthvalue, and the truthvalue of the whole sentence is determined by the fact that the interpretation is a boolean valuation. In any boolean valuation, certain manipulations preserve logical equivalence; for example, one can distribute disjunction through conjunction, and obtain

$$((Pa \vee \neg Q) \wedge (Pa \vee [S_1, t_1]\exists x.Px)) \vee \langle S_2 : \exists x.Px \rangle Q \quad . \tag{10.2}$$

This sentence has the same truthvalue as (10.2). Applying the distribution rule again, we arrive at the CNF sentence

$$\begin{aligned}&((Pa \vee \neg Q \vee \langle S_2, t_2 : \exists x.Px \rangle Q) \\ &\qquad \wedge (Pa \vee [S_1, t_1]\exists x.Px) \vee \langle S_2 : \exists x.Px \rangle Q)\end{aligned} \tag{10.3}$$

The rules in Robinson [59], pages 150–152, are sufficient to convert any boolean sentence into CNF. Since the rules respect boolean valuations, the CNF sentence is equivalent to the original sentence if their interpretations are boolean valuations. This yields the following theorem.

THEOREM 10.1. *A set of sentences of a propositional modal language is valid if and only if its clause form is.*

For propositional languages, we define the following notion of subsumption.

DEFINITION 10.3. *A clause C* subsumes *a clause D if and only if every literal in C is also in D. A set of clauses W* dominates *a set of clauses W' if and only if every clause in W' is subsumed by a clause in W.*

The following theorems about dominated clause sets will be useful in proving termination of proof procedures.

THEOREM 10.2. *Subsumption and domination are transitive relations.*

Proof. Let C subsume C', and C' subsume C''. Consider an arbitrary literal L of C. By assumption, L is in C', and hence in C'' as well. Thus C subsumes C''.

Let W dominate W', and W' dominate W''. Consider an arbitrary clause C'' of W''. This clause is subsumed by a clause C' of W', which, in turn, is subsumed by a clause C of W. Because subsumption is transitive, C'' is subsumed by C, and W dominates W''.▮

THEOREM 10.3. *Let Z be a finite set of literals, and $\sigma_1, \sigma_2, \ldots$ a sequence of clause sets constructed using just these literals. In every such infinite sequence, some σ_i will dominate some σ_j for $i < j$.*

Proof. Let $L_1 \ldots L_n$ be the n literals of Z. Consider the power set $\mathcal{P}(Z)$ of Z, which contains 2^n members. For each member we can construct a clause by inserting disjunctions between the literals; let us call these the *canonical clauses* $C(Z)$ of Z. These are really the only distinct clauses that can be constructed from Z, in the sense that every clause will subsume and be subsumed by one of these canonical clauses. Now consider the power set of $\mathcal{P}(C(Z))$ of $C(Z)$, which has 2^{2^n} members. Any clause set σ_i can be transformed into a member W of $\mathcal{P}(C(Z))$ by replacing each of its clauses with the equivalent canonical clause. σ_i both dominates and is dominated by W. In any sequence $\sigma_1, \sigma_2, \ldots$ longer than 2^{2^n}, there must be some σ_i and σ_j that can be transformed into the same set W of $\mathcal{P}(C(Z))$. By the transitivity of domination, these clause sets dominate each other.▮

Domination is a syntactic concept, but is related to the semantic concept of satisfiability by the following theorem.

THEOREM 10.4. *If W dominates W', then W is unsatisfiable only if W' is.*

Proof. Assume that W is satisfiable, so that every clause in W is true in an interpretation. Since W dominates W', every clause of W' is subsumed by some clause of W, and hence also true in that interpretation. Thus W' must also be satisfiable.▮

In what follows, we will need the definitions of detached and pure literals, and tautologous clauses.

DEFINITION 10.4. *Let W be a finite set of clauses that contains a literal L, and let A be the set of literals of W other than L. L is* semantically detached in W *if, for every interpretation I of A, there is an interpretation of $A \cup \{L\}$ that agrees with I on A, and for which L is true.*

L is pure in W *if there is no occurrence of its complement.*

A clause is a tautology *if it contains both a literal and its complement.*

The notion of semantic detachment is important because it picks out literals whose interpretation is, in a sense, independent of the rest of the literals of W. Consider the set W' obtained from W by deleting all clauses that contain the detached literal L. W' is satisfiable if W is, since it is a subset of W. Suppose W' is satisfied by some interpretation I'. Since L is detached, we can extend I' to an interpretation I in which L is true, and hence all clauses containing L are also true in I, so that I satisfies W. The satisfiability of W is thus equivalent to that of W', so that clauses containing L can be disregarded in determining the satisfiability of W. We have the following theorem.

THEOREM 10.5. *If L is a literal detached in a set of clauses W, then the subset $W' \subset W$ obtained by deleting all clauses containing L is satisfiable if and only if W is.*

The definition of a literal pure in a set of clauses is a syntactic one, but is closely related to the semantic concept of a detached literal. Certainly purity is a necessary condition for detachment; if a detached literal L were not pure, the atom set A would contain $\sim L$, and an interpretation that made $\sim L$ true could not also make L true. Purity is also a sufficient condition when the interpretations of W are *all* boolean valuations. Suppose L is pure in W, so that its complement $\sim L$ does not appear. Let I be a boolean valuation of the literal set A. Since A does not contain L or its complement, we can extend I to an interpretation in which L is true. We conclude that, if the interpretations of W are all boolean valuations, a literal being pure in W is equivalent to its being detached in W.

10.1.2 A Tableau Method for Generalized Davis-Putnam Rules

The Davis-Putnam method for the propositional calculus consists of four rules that rewrite a set of clauses into a smaller set or sets equivalent in unsatisfiability to the original. These rules make essential use of the fact that all boolean valuations are allowed interpretations. Of course, not every boolean valuation is an interpretation of propositional modal logics, because some of the belief literals can conflict, even though they are not complementary. We will generalize the Davis-Putnam rules so that they can take into account more restrictive interpretations of the literals. We express these generalized rules in terms of a tableau system DP. It is convenient to use DP to arrive at proof-theoretic properties; from it we can derive a decision procedure.

In what follows, we assume that W is a finite set of clauses, and L is a literal contained in a clause of W. C is an arbitrary clause (not necessarily in W).

DEFINITION 10.5. *The system* DP *has the following postulates.*

Empty: $\blacksquare, W \Rightarrow$

Unsat: $W \Rightarrow$, *if W contains an unsatisfiable literal set.*

Valid: $\dfrac{C, W \Rightarrow}{W \Rightarrow}$, *where C is valid.*

False: $\dfrac{L \vee A, W \Rightarrow}{A, W \Rightarrow}$, *where L is always false.*

Single: $\dfrac{L, W \Rightarrow}{L, W' \Rightarrow}$, *where L is a singleton clause such that either L or $\sim L$ occurs in W; W' is obtained from W by deleting all clauses containing L, and deleting all occurrences of $\sim L$ from all remaining clauses of W.*

Detach: $\dfrac{W \Rightarrow}{W' \Rightarrow}$, *where L is a literal detached in W, and W' is obtained from W by deleting all clauses containing L.*

Split: $$\frac{W \Rightarrow}{\sim L, W^{+}, W' \Rightarrow \qquad L, W^{-}, W' \Rightarrow},$$

where W' is the set of clauses of W not containing L or $\sim L$, W^{+} the set obtained by deleting all occurrences of L from those clauses of W containing such an occurrence, and W^{-} the corresponding set for $\sim L$.

The null clause is false in every interpretation, and so we have the axiom *Empty*. The axiom *Unsat* is used when W contains a set of literals (singleton clauses) that can be shown to be unsatisfiable.

The rule *Valid* allows us to eliminate clauses that are always true; in the propositional case, these are the tautologies, but, in the case of fully or partially interpreted languages, clauses other than tautologies may also be always true. The rule *False* is similar to *Valid*, but checks that a literal is falsified by every model; if so, it can be removed from its clause. The rule *Single* removes singleton clauses, and also eliminates any clause in which the singleton literal appears, and any occurrence of its complement. *Detach* also deletes clauses, given a literal that is detached.

The above rules are very important, because they allow us to prune clauses from W without the expense of splitting. There are cases, however, in which splitting is unavoidable, and the rule *Split* must be invoked.

For any given language, we can particularize the postulates of DP according to the interpretations of the language. For the rules *Valid*, *False*, and *Detach*, we may give sufficient conditions for validity of clauses, unsatisfiability of literals, and detachment of literals. These sufficiency conditions are optional, and serve to increase the efficiency of derivations. In order to use DP as a decision procedure, however, necessary and sufficient conditions under which a literal set is unsatisfiable must be specified (the axiom *Unsat*), and these conditions must be effectively computable.

DP simplifies to the Davis-Putnam rules for the propositional calculus when the interpretations are all boolean valuations. Under these interpretations, a literal set is unsatisfiable if and only if a literal and its complement appear, yielding an effective means of determining if the *Unsat* axiom applies. A clause C is valid *iff* it is a tautology, so the rule *Valid* is just the elimination of tautologies. There are no special interpretations for predicates, and the *False* rule does not apply. The *Detach* rule is equivalent to the pure-literal rule of the Davis and Putnam method in the propositional case, since we have argued that semantic detachment and purity coincide for propositional interpretations. *Single* is similar to the one-literal rule, but differs in that the literal L is added in the bottom sequent. Since this sequent is pure in L, L can be removed by the application of the rule *Detach*. Finally, the splitting rule of Davis and Putnam is the same as *Split*.

The rules of DP have as their intent the breakdown of clauses into their equivalent atom sets. Of course, we could do this by repeated applications of the rule *Split*, but this would be computationally inefficient, since n clauses of two different literals each would yield 2^n atom sets. All the rules except *Split* are designed to reduce the number of disjunctions without splitting.

Example. Let $s = (P \supset P) \wedge (P \supset R) \wedge (R \supset Q) \wedge (Q \supset P)$. The clause form of s is $P \vee \neg P,\ R \vee \neg P,\ Q \vee \neg R,\ P \vee \neg Q$.

$$\textit{Split}\;\dfrac{\textit{Valid}\;\dfrac{P \vee \neg P, R \vee \neg P, Q \vee \neg R, P \vee \neg Q \Rightarrow}{R \vee \neg P, Q \vee \neg R, P \vee \neg Q \Rightarrow}}{\textit{Single}\;\dfrac{R, Q \vee \neg R, P \Rightarrow}{R, Q, P \Rightarrow} \qquad \textit{Single}\;\dfrac{Q \vee \neg R, \neg Q, \neg P \Rightarrow}{\neg R, \neg Q, \neg P \Rightarrow}} \tag{10.4}$$

It is an interesting property of the rules DP that we can always construct a tableau whose branches terminate in literal sets that are logically equivalent to the initial clause set. Whether these literal sets are satisfiable or not is a property of the particular interpretation we choose for the language. Given an effective method of determining satisfiability of literal sets in the rule *Unsat*, we have a decision procedure.

In the example above, the tableau is open because each terminal node is satisfiable, and we can use it to construct a propositional model of s. Consider the literal set of the left branch. We can construct an interpretation that makes all the literals of this set true, and this interpretation satisfies s. A similar construction for the literal set of the right branch also yields a model of s.

We now prove an important fact about the system DP: that each rule preserves validity—the top sequent is valid exactly when the bottom sequents are. This is a stronger result than soundness, and we can use it to establish a decision procedure.

THEOREM 10.6. *The top sequent of each rule of* DP *is valid if and only if the bottom sequents are.*

Proof. For the rule *Valid*, if C is valid, then its truthvalue is t in all interpretations. Hence $C, W \Rightarrow$ is valid exactly when W is unsatisfiable, *i.e.*, when $W \Rightarrow$ is valid.

For the rule *False*, if a literal L is false in every model, then the clause $L \vee A$ is equivalent to the clause A.

For the rule *Single*, there are two cases. If L has value t in an interpretation, then any clause containing L also has value t. Hence all these clauses may be eliminated from W without affecting the truthvalue of the sequent. Also, any clause C that contains a literal $\sim L$ is equivalent to the clause C' obtained by deleting that literal. The sequent $L, W' \Rightarrow$ thus has the same truthvalue as $L, W \Rightarrow$ when L is true. If L is false, both top and bottom sequents are obviously true. Hence both top and bottom sequents have the same value in any interpretation.

For the rule *Detach*, the equivalence of validity of the sequents follows immediately from Theorem 10.5.

For the rule *Split*, suppose the top sequent is true in an interpretation. If one of the clauses in W' is false, then both bottom sequents are also true. If one of the clauses in W^+ is false, then L must also be false, because all clauses of W^+ contain L; so both bottom sequents are true. And if one of the clauses of W^- is false, $\sim L$ must be false, so both bottom sequents are again true.

Suppose the two bottom sequents are true in an interpretation. If one of the clauses of W', W^+, or W^- is false, then the top sequent is true. The only other way both bottom sequents could be true is if L and $\sim L$ are both false, an impossibility.∎

As an immediate corollary, we have that the rules *DP* are sound. We can use the Theorem 10.6 to establish a decision procedure.

We stated above that it was a property of the system **DP** that a finite tableau could be constructed, such that all its branches ended in literal sets. We now show that this is so. To construct a tableau of the requisite sort, we apply the rules *DP* in any order to the initial clause set. Each rule application produces a sequent (or sequents, in the case of *Split*) whose disjunction count is less than that of the premise sequent. For example, the application of the rule *Split* in (10.4) above produced sequents each of which had a disjunction count of 1, while the premise sequent had 3 disjunctions. So if we can keep applying the rules, we will eventually generate a tableau whose leaf sequents have disjunction counts of 0, *i.e.*, contain only singleton clauses. We need to show that it is always possible to apply some rule of *DP* whenever a disjunction exists. Consider a sequent $C, W \Rightarrow$, where C consists of the literals $P_1 \vee P_2 \ldots$ It is always possible to apply the rule *Split*, with $L = P_1$.

Finally, if a leaf node of a tableau is a literal set, then we can apply the rule *Unsat* to close the branch if the literal set is unsatisfiable. Otherwise, because of Theorem 10.6, we know that the root sequent is not valid, *i.e.*, the original clause set W is satisfiable. We collect these results about the system **DP** in the following theorem:

THEOREM 10.7. *Let W be a finite set of clauses. By applying the rules DP to an initial sequent $W \Rightarrow$ in any order, we generate a finite tableau whose open leaf nodes are of the form $L_1, L_2, \ldots \Rightarrow$, where the literal set $\{L_1, L_2, \ldots\}$ is satisfiable. The set W is satisfiable just in case there exists one such open branch.*

Note that the proof of this theorem only requires the use of the postulates *Split*, *Unsat*, and *Empty*. To apply the rules for an arbitrary language, we only need specify an effective procedure for computing *Unsat*. Sufficient conditions for detachment of literals and validity of clauses can make the construction more efficient, however. One useful heuristic is to delay the use of *Split* as much as possible in favor of *Detach*, *Single*, and *Valid*, since any order of application of the rules generates the required tableau.

10.1.3 The Procedure DP

We now describe a total procedure, $DP(W)$, that returns **Sat** if the clause set W is satisfiable, and **Unsat** if it is not. $DP(W)$ essentially searches a particular tableau for W. The concept of a *reduction of a clause set W* is needed in the definition. If one of the rules *Single, Valid, False*, or *Detach* can be applied to the sequent $W \Rightarrow$, we say that W is *reducible*. If $W' \Rightarrow$ is the sequent that results from the application of the rule, we write $reduce(W) =_{df} W'$.

DEFINITION 10.6. *Let $UNSAT(Z)$ be a total function that returns* t *if the literal set Z is unsatisfiable, and* f *otherwise. The recursive function $DP(W)$ is defined as follows.*

```
1.  While $W$ is reducible do
          $W \leftarrow reduce(W)$
          if $\blacksquare \in W$ then return Unsat

2.  If there is a nonsingleton clause $C$ of $W$ whose first lit-
    eral is $L$,
          if $DP(\{\sim L, W^+, W'\})$ = Unsat and
             $DP(\{L, W^-, W'\})$ = Unsat
          then return Unsat
```

```
        else return Sat

3.  If $UNSAT(W) = \mathrm{t}$
        then return Unsat
        else return Sat
```

It is easy to show that *DP*(W) terminates with **Unsat** if there is a closed tableau for $W \Rightarrow$, and terminates with **Sat** if there isn't. While *DP*(W) is in step (1), it is following a tableau constructed from $W \Rightarrow$ by applying the rules *Single*, *Valid*, and *Detach*, which do not branch. If at some point the tableau closes with the axiom *Empty*, *DP*(W) halts with value **Unsat**. When there are no more reduction rules to apply, either W is a set of singleton clauses, in which case *UNSAT* is called and **Unsat** is returned exactly if the tableau closes; or there is at least one nonsingleton clause C. In the latter case, *DP* follows the two branches of the tableau produced by the *Split* rule, calling itself recursively with the appropriate clause sets. If each of these branches closes, the recursive calls will return **Unsat**, and so will *DP*(W); otherwise it will return **Sat**. Because the tableau constructed by the rules *DP* is always finite (Theorem 10.7), *DP*(W) will always halt.

10.2 The Method for Unquantified B^+

To formulate the decision procedure of Definition 10.6 for unquantified B^+, we need to do two things: define sufficient conditions for a literal of B^+ to be detached, and give a decision procedure *UNSAT*(Z) for literal sets Z. This latter condition amounts to having belief derivation ($\Gamma \mathrel{\natural\!\!\vdash} p$) be decidable for a particular logic of B^+, as we now show. The definition of $Y(S_i, t_j, Z)$ used here is given in Section 5.5; it is the set of sentences that are stated by the literal set Z to be in the belief set of agent S_i at time t_j, including common beliefs.

THEOREM 10.8. *Let Z be a finite set of literals of B^+. Define a function $UNSAT(Z) \mapsto \{\mathrm{t}, \mathrm{f}\}$ by*

1. *If an ordinary literal and its complement are in Z, then $UNSAT(Z) = \mathbf{t}$.*
2. *If for any negative belief literal $\neg[S_i, t_j]p$ it is true that $Y(S_i, t_j, Z) \Vdash_{\rho(i)} p$, then $UNSAT(Z) = \mathbf{t}$.*
3. *If for any positive circumscription literal $\langle S_i : \Gamma\rangle p$ it is true that $\Gamma \nVdash_{\rho(i)} p$, then $UNSAT(Z) = \mathbf{t}$.*
4. *If for any negative circumscription literal $\neg\langle S_i : \Gamma\rangle p$ it is true that $\Gamma \Vdash_{\rho(i)} p$, then $UNSAT(Z) = \mathbf{t}$.*
5. *If none of the above conditions hold, $UNSAT(Z) = \mathbf{f}$.*

If $\Vdash_{\rho(i)}$ is decidable for every agent S_i, then $UNSAT(Z)$ is a total function that returns t *exactly when Z is unsatisfiable.*

Proof. Suppose none of the first four conditions hold. By the completeness of B$^+$, there is a model that satisfies Z. On the other hand, if one of the first four conditions holds, there is a closed tableau for $Z \Rightarrow$ using the rules of B$^+$ in Definition 5.1, and so Z is unsatisfiable because B$^+$ is sound.∎

10.2.1 Detachment and Validity for B$^+$

We now derive criteria for detachment and validity that can be used in the rules *Valid*, *False*, and *Detach*. For the former two rules, we consider circumscriptive literals. From the definition of the valuation function for these literals (Section 5.2.1), it is obvious that $\langle S_i, \Gamma\rangle p$ is valid if $\Gamma \Vdash_{\rho(i)} p$, and unsatisfiable if not. That is, we can determine the truthvalue of any circumscriptive literal by simply computing the relevant belief derivation. Using the rules *Valid* and *False*, all occurrences of circumscriptive literals can be eliminated from the clause set.

As for detachment, in the case of ordinary (nonmodal) literals it is equivalent to purity, because the interpretations of these literals are all boolean valuations. We now give a sufficient and effectively computable condition, namely *nonopposition*, for belief atoms to be detached in a clause set.

DEFINITION 10.7. *A positive belief literal $[S_i, t_j]p$ is* unopposed in a clause set *W iff W contains no literal of the form $\neg[S_n, t_k]q$ for $t_k \geq t_j$*

and $i = 0$ *or* $i = n$. *A negative belief literal* $\neg[S_n, t_k]q$ *is unopposed in* W *iff* W *contains no literals of the form* $[S_i, t_j]p$ *for* $t_k \geq t_j$ *and* $i = 0$ *or* $i = n$.

Nonopposition is a syntactic property of a belief literal with respect to a set of clauses. It is a sufficient condition for detachment, although not a necessary one: for example, the atom $[S_i, t_k]p$ is opposed to $\neg[S_i, t_k]q$, yet it also detached if $p \not\vdash_{\rho(i)} q$. We now prove that nonopposition implies detachment.

THEOREM 10.9. *If a clause set* W *has a* B^+*-model, then it has one in which every unopposed literal is true.*

Proof. Consider first the positive belief literal $[S_i, t_j]p$, where $i \neq 0$ (*i.e.*, this is not a common belief). It is easy to see that this atom will be detached in a clause set W if there are no literals of the form $\neg[S_i, t_k]q$, where $t_j \leq t_k$. If there is a B^+-model m of W, we can always construct a model m' in which $[S_i, t_j]p$ is true. To do this, simply take all the atoms $[S_i, t_k]\gamma$ and $[S_0, t_k]\delta$ of W for $t_k \leq t_j$, and form the deduction structure $d_i^j =_{\mathrm{df}} \langle \Gamma \cup \Delta, \rho(i) \rangle$ (note that p is one of the γ). If d_i^j is substituted for the deduction structure of agent S_i in submodel m_j of m, then the resulting model m' satisfies W (because there are no negative atoms of the form $\neg[S_i, t_k]r$ for $t_j \leq t_k$) and $[S_i, t_j]p$.

If $i = 0$, then for every agent S_n and every time t_k we can form a deduction structure d_n^k as above, and it is easy to show that the model m' constructed using d_n^k in m_k satisfies W if m does.

For the negative literal $\neg[S_i, t_j]p$, we form m' from m by substituting the deduction structure $\langle \emptyset, \rho(i) \rangle$ for agent S_i and $\langle \emptyset, \rho(0) \rangle$ for S_0 at every time previous or equal to t_j. m' satisfies W because there are no positive belief literals for S_i or S_0 up to or at the time t_j, and also obviously satisfies $\neg[S_i, t_j]p$.∎

Using this result, we have the following sufficient conditions for detachment in B^+.

THEOREM 10.10. *A literal* L *of* B^+ *is detached in a clause set* W *if it is an ordinary literal pure in* W, *or a belief literal unopposed in* W.

10.3 The Method for Unquantified B_s^+

At this point we make the choice of a nonintrospective interpretation A_K^+ of A^+, which gives a logic family BK^+ that is an extension of BK. In the family of logics BK, the belief deduction operator $\not\vdash$ is replaced by provability in a sequent system. In a similar manner, the definition of *UNSAT* (10.8) can be modified by substituting a recursive call to $DP(W)$ for every instance of belief derivation. It is here that termination conditions prove important, because the procedure thus generated could cycle infinitely on the same argument W. Thus we define a modification of $DP(W)$, $DPP(W, \Sigma)$, in which the second argument Σ is used to decide when a call to *DPP* is subsumed by some previous call.

We now define *DPP*, establish its termination, and prove that it is a decision procedure for the saturated logic BK_s^+. We will also give a short example proof done by a computer implementation of the procedure; a more extensive example solving the hard form of the Wise Man Puzzle (showing ignorance) is given in the appendix.

DEFINITION 10.8. *Let Z be a finite set of literals of propositional BK_s^+, and Σ be a set each of whose members σ_i is a clause set of BK_s^+. Define a function $UNSAT(Z, \Sigma) \mapsto \{\mathbf{t}, \mathbf{f}\}$ by*

1. *If an ordinary literal and its complement are in Z, then $UNSAT(Z, \Sigma) = \mathbf{t}$.*
2. *Let $\neg[S_i, t_j]p$ be any negative belief literal of Z, and W the clause form of $Y(S_i, t_j, Z) \cup \{\neg p\}$. If W is not dominated by any member of Σ, and $DPP(W, \Sigma \cup \{W\}) =$ **Unsat**, then $UNSAT(Z, \Sigma) = \mathbf{t}$.*
3. *Let $\langle S_i : \Gamma\rangle p$ be any positive circumscription literal of Z, and W the clause form of the sentences Γ and $\neg p$. If $DPP(W, \emptyset) =$ **Unsat**, then $UNSAT(Z, \Sigma) = \mathbf{t}$.*
4. *Let $\neg\langle S_i : \Gamma\rangle p$ be any negative circumscription literal of Z, and W the clause form of Γ and $\neg p$. If $DPP(W, \emptyset) =$ **Sat**, then $UNSAT(Z, \Sigma) = \mathbf{t}$.*
5. *If none of the above conditions hold, $UNSAT(Z) = \mathbf{f}$.*

Remarks. The only change from Definition 10.8 is that the belief operator has been replaced by a call to *DPP* in the second, third, and fourth items. The second item contains the recursive call to *DPP* that mimics the attachment rule A_K^+; we could easily change this to reflect an introspective interpretation of A^+, parallelling the development of BS4 and BS5.

Note that we have added the condition that W not be dominated by any member of Σ in the second item. This condition is important to prevent infinite recursion of *UNSAT* and *DPP*. For example, suppose $Z = \{[S_0, t_0]\neg[S_1, t_1]p, \neg[S_1, t_1]p\}$, so that $W = \{[S_0, t_0]\neg[S_1, t_1]p, \neg[S_1, t_1]p\} = Z$ in the second item. If it were not for the domination check, *DPP* and *UNSAT* would keep calling each other with the same first argument. The second argument to both these function accumulates the sets of clauses that have already been used as arguments to *DPP*. If a call to *DPP* would involve a clause set W that is dominated by a clause set $W' \in \Sigma$, this call is ignored. The end result is that *UNSAT* and *DPP* are effectively computable, as we now prove.

THEOREM 10.11. *UNSAT*(Z, Σ) *terminates for all values of its arguments.*

Proof. Let Z' be the set of all literals that are constructable from the belief operators and predications that appear in Z; Z' is finite because Z is. Every recursive call to *DPP* or *UNSAT* that is made during the computation of *UNSAT*(Z, Σ) will involve only literals from Z'. Now assume that the computation is nonterminating, so that there is some branch of the computation that involves an infinite number of calls to *UNSAT*. There are two cases to consider. If the infinite branch goes through item (2) an infinite number of times, then the set Σ grows an unbounded amount. By Theorem 10.3, however, for some finite Σ there must be a member σ_i that dominates some other member σ_j. This violates the conditions of item (2), and so no such infinite computation is possible.

If the infinite branch doesn't go through item (2) an infinite number of times, then it must go through either item (3) or (4) an infinite number of times. But this is also impossible, since every recursive call in these items reduces the number of circumscription operators by at least one, and hence cannot generate an infinite branch.∎

We now prove that Definition 10.8 yields a decision procedure for the logic BK_s^+.

THEOREM 10.12. *Let s be a sentence of B^+. s is a theorem of BK_s^+ if and only if $DPP(W,\emptyset) =$ **Unsat**, where W is the clause form of s.*

Proof. First part: if $DPP(W,\emptyset) =$ **Unsat**, then $\vdash_{\mathsf{BK}_s^+} \Rightarrow s$. The method for proving this is to show that the tableau rules DP (Definition 10.5) are admissible in BK_s^+, then convert the trace of the computation of DPP into a tableau in the system $\mathsf{DP}^+ =_{\mathrm{df}} \mathsf{DP} + A_K^+ + Circ_1 + Circ_2$.

Second part: if $\vdash_{\mathsf{BK}_s^+} \Rightarrow s$, then $DPP(W,\emptyset) =$ **Unsat**. The method for proving this has three steps.

1. Any closed tableau in BK_s^+ can be converted into a closed tableau in DP^+.
2. Any closed tableau in DP^+ can be transformed into one in which there is no branch containing subsumed A_K^+ clauses.
3. The decision procedure DPP follows this tableau.

We leave the details for the reader.∎

Example. We illustrate the use of the decision procedure on the following set of sentences of propositional B. This example is a cleaned-up trace of a program that implements the procedure, without the subsumption check in item (2) of the definition of *UNSAT*.

The language has two agents, `s1` and `s2`, as well as "any fool" `F`. There are two times with the ordering `t1` $<$ `t2`, and two primitive propositions `p1` and `p2`. The axioms are as follows:

```
W1:    [F,t1](p1 ∨ p2)
W2:    [s1,t1]¬p1
L1:    [s1,t1]¬p1 ⊃ [F,t2]¬p1
L2:    [s1,t1]p1 ⊃ [F,t2]p1
```

W1 states that it's a common belief at time **t1** that either **p1** or **p2** is true. From W2, **s1** believes **¬p1** at time **t1**. The axioms **L1** and **L2** describe the transfer of beliefs from **t1** to **t2**: whatever **s1** initially believes about **p1**, anyone (including **s2**) will believe at **t2**.

We wish to prove that **s2** believes **p2** at time **t2**, given the above axioms. To do this, we convert the axioms into clauses, and add to them **¬[s2,t2]p2**, which is the negation of what we wish to prove. These clauses are given to the procedure *DPP* as the input clause set W (see Definition 10.6), and a proof is obtained if *DPP* returns **Unsat**. A trace of the resulting computation follows. Each time a reduction rule is applied to W, the type of rule is indicated, and the resultant reduced clause set *reduce*(W) is printed. When a split occurs, or the unsatisfiability of a modal literal set is being checked, a recursive call is made to *DPP*; all calls are numbered so that it is possible to tell when the call is finished, and what its result was.

```
1. Entering DPP level ()
 GOAL:     ¬[s2,t2]p2
 W1:       [F,t1](p1 ∨ p2)
 W2:       [s1,t1]¬p1
 L1:       ¬[s1,t1]¬p1 ∨ [F,t2]¬p1
 L2:       ¬[s1,t1]p1 ∨ [F,t2]p1
```

We enter the procedure *DPP* with the clause form of the axioms, and the negation of the sentence to be proven (this is the GOAL clause). The procedure first applies all nonsplitting reductions.

```
Single rule:  W2:      [s1,t1]¬p1

 L2:       ¬[s1,t1]p1 ∨ [F,t2]p1
 L1:       [F,t2]¬p1
 W1:       [F,t1](p1 ∨ p2)
 GOAL:     ¬[s2,t2]p2
 ----
 W2:       [s1,t1]¬p1
```

The *Single* rule applies, using the singleton literal of **W2**. The resulting reduced clause set is printed out. No clauses are eliminated, but one of the literals of **L1** is erased. **W2** is drawn below a dashed line to indicate that it has already been considered by the *Single* rule.

```
Single rule:  L2:      [F,t2]¬p1
Single rule:  W1:      [F,t1](p1 ∨ p2)
Single rule:  GOAL:    ¬[s2,t2]p2

 L2:       ¬[s1,t1]p1 ∨ [F,t2]p1
 ----
 W1:       [F,t1](p1 ∨ p2)
 L1:       [F,t2]¬p1
 W2:       [s1,t1]¬p1
 GOAL:     ¬[s2,t2]p2
```

The *Single* rule applies three more times; we have printed out the reduced clause set at the end of the third application. No clauses or literals have been eliminated; the four singleton clauses are drawn below the dashed line to indicate that they have already been considered by for use in the *Single* rule.

DPP has exhausted its use of the *Single* rule, and now tries the *Detach* rule. It doesn't apply, because the presence of the GOAL literal opposes every belief literal of agent s2 and F, and the first literal of L2 opposes the literal of W2. *DPP* next applies the splitting rule, creating two recursive calls to itself.

```
Splitting on literal ¬[s1,t1]p1

 2. Entering DPP level ()

  SPLIT:    ¬[s1,t1]p1
  L2:       ¬[s1,t1]p1 ∨ [F,t2]p1
  ----
  W1:       [F,t1](p1 ∨ p2)
  L1:       [F,t2]¬p1
  W2:       [s1,t1]¬p1
  GOAL:     ¬[s2,t2]p2
```

This is one branch of the split that is based on the first literal of L2. Note that we keep the belief literals that have already been considered by the *Single* rule.

```
 Single rule:  SPLIT:   ¬[s1,t1]p1

  ----
  SPLIT:    ¬[s1,t1]p1
  W1:       [F,t1](p1 ∨ p2)
  L1:       [F,t2]¬p1
  W2:       [s1,t1]¬p1
  GOAL:     ¬[s2,t2]p2

 At level () with the following negative modal atoms:
  SPLIT:    ¬[s1,t1]p1
  GOAL:     ¬[s2,t2]p2
   .
```

The *Single* rule eliminates clause L2. At this point there are no more reduction rules or splits that apply, and we have a belief literal set, of which two are negative. By item (2) of Definition 10.8, *DPP* must be called recursively on each of these in turn; if either one is unsatisfiable, then this branch of the split is unsatisfiable. We need not keep track of subsumption for this particular proof, because there are no negative belief literals under the scope of a common belief operator that could cause an infinite recursion.

```
Attaching to modal atom SPLIT:   ¬[s1,t1]p1

  3.  Entering DPP level ((s1 t1))

  W1:       [F,t1](p1 ∨ p2)
  GOAL:     ¬p1
  W1:       p1 ∨ p2
  W2:       ¬p1
```

We have now attached to agent `s1`'s view at time `t1`. The original belief literal `W1` has generated two clauses because it is a common belief. The clause `GOAL` and `W2` are agent `s1`'s beliefs.

```
  Single rule:  W2:       ¬p1
  Single rule:  W1:       p2
  Single rule:  GOAL:     ¬p1
  Single rule:  W1:       [F,t1](p1 ∨ p2)

   ----
   W1:      [F,t1](p1 ∨ p2)

  Unopposed literal in W1:      [F,t1](p1 ∨ p2)

 3.  Satisfied!  at level ((s1 t1))

At level () Satisfied modal attachment to SPLIT:   ¬[s1,t1]p1
```

The *Single* rule applies enough times to eliminate all the ordinary literals. The only clause left is the singleton `W1`. This literal is unopposed, and so the *Detach* rule eliminates it. The clause set becomes empty, so this particular call to *DPP* returns the answer **Sat**. The negative belief atom (`¬[s1,t1]p1`) that generated this attachment is thus satisfied, and *DPP* now checks the other negative belief atom.

```
Attaching to modal atom GOAL:   ¬[s2,t2]p2

  4.  Entering DPP level ((s2 t2))
```

```
 L1:      [F,t2]¬p1
 W1:      [F,t1](p1 ∨ p2)
 GOAL:    ¬p2
 W1:      p1 ∨ p2
 L1:      ¬p1

Single rule:  L1:      ¬p1
Single rule:  W1:      p2

 GOAL:    ■
 W1:      [F,t1](p1 ∨ p2)
 L1:      [F,t2]¬p1
 ----

4.  UnSatisfied!  at level ((s2 t2))

At level () Failed modal attachment to GOAL:     ¬[s2,t2]p2

2.  UnSatisfied!  at level ()
```

The *Single* rule finds the contradiction in s1, L1, and GOAL, generating an empty clause. The negative belief atom is thus unsatisfied, and hence so is this branch of the split. *DPP* now checks the other branch.

```
5.  Entering DPP level ()

 SPLIT:   [s1,t1]p1
 L2:      ¬[s1,t1]p1 ∨ [F,t2]p1
 W1:      [F,t1](p1 ∨ p2)
 L1:      [F,t2]¬p1
 W2:      [s1,t1]¬p1
 GOAL:    ¬[s2,t2]p2

Single rule:  SPLIT:    [s1,t1]p1
Single rule:  L2:       [F,t2]p1
Unopposed literal in SPLIT:    [s1,t1]p1
Unopposed literal in W2:       [s1,t1]¬p1

At level () with the following negative modal atoms:

 GOAL:    ¬[s2,t2]p2
```

```
Attaching to modal atom GOAL:    ¬[s2,t2]p2

 6.  Entering DPP level ((s2 t2))

  L1:        [F,t2]¬p1
  W1:        [F,t1](p1 ∨ p2)
  L2:        [F,t2]p1
  GOAL:      ¬p2
  L2:        p1
  W1:        p1 ∨ p2
  L1:        ¬p1

 Single rule:  L1:       ¬p1

  W1:        p2
  L2:        ■
  GOAL:      ¬p2
  L2:        [F,t2]p1
  W1:        [F,t1](p1 ∨ p2)
  L1:        [F,t2]¬p1
  ----

 6.  UnSatisfied!  at level ((s2 t2))

At level () Failed modal attachment to GOAL:  ¬[s2,t2]p2

4.  UnSatisfied!  at level ()

1.  Unsatisfied!  from split at level ()
```

In this branch of the split, reduction rules apply until there is a single negative belief literal left. Attaching to this literal generates call 6 to *DPP*, and an empty clause is produced. Hence the negative belief literal is unsatisfiable, and so is this branch of the split (call 4 to *DPP*). Finally, the results from both branches of the split are collected, and the first call to *DPP* returns unsatisfiable, indicating that `[s2,t2]p2` is provable from the axioms.

11. Herbrand's Theorem for qB with Functions

For logics that make essential use of first-order quantification, there is no procedure that can decide whether a given sentence is a theorem of the logic. However, there are procedures that are *complete*, in the sense that if a sentence actually is a theorem, the procedure will show this. From the results of Chapter 9 we know that the logic **qB** has such a procedure, based on the method of block tableaux. While this proof method is adequate for analyzing the properties of **qB**, it is not suitable for automatic theorem-proving. The reasons this is so are basically threefold: the splitting caused by propositional rules such as D_2 and C_2; the many choices for instantiation of quantified variables; and the computational requirements of belief derivation in the attachment rule, which essentially acts like a recursive call to the proving procedure. Of these reasons, the first two are common to first-order logic as well; early attempts at automatic theorem-proving foundered because of them (see the historical notes in Chang and Lee [5], pages 62 ff.).

The first great success in automatic methods was achieved by Robinson [60], using a technique called *resolution*, which addressed both the splitting and variable instantiation problems. Robinson's technique was based on results that had been obtained by Herbrand relating the unsatisfiability of a sentence in a certain form to a finite set of its ground instances (Herbrand's Theorem). However, Herbrand's result was a theorem about first-order logic, and does not carry over in a straightforward manner to quantified-in modal logics; at least one recent attempt to develop an analog for modal logics has had negative results (see Haspel [17]). Because of this, AI researchers whose systems made use of modal logics of knowledge and belief did not have available *any* automatic proof methods for the logic. As an alternative, they developed the indirect technique of axiomatizing the possible-world semantics of modal logic in a first-order language, for which automatic methods had been developed (see Haspel [17], McCarthy *et al.* [43], and Moore [51]). However, the indirection of axiomatization makes the proof methods less efficient and harder to understand, so there is still a great need for efficient direct procedures for modal logic.

This chapter lays the foundation for resolution methods applicable to the quantified-in logic qB, by extending Herbrand's Theorem from first-order logic to qB. Because of the correspondence property, this result also holds for quantified modal logics with possible-world semantics. In the next chapter we utilize Herbrand's Theorem to arrive at resolution methods for qB, and show how resolution can be specialized to yield AI rule-based systems.

The development in this chapter is similar to that in Robinson [59] and Chang and Lee [5], but, of necessity, the definitions and theorems are modified to take into account the special semantics of modal atoms. First we define an extended language $L^{\mathsf{qB}^\bullet}$ that is similar to L^{qB} but allows functional terms in addition to constants. A *skolem normal form* for $L^{\mathsf{qB}^\bullet}$ is derived in which existential quantifiers are eliminated in favor of newly introduced skolem functions. A restricted class of B-models, the *normative Herbrand models*, prove sufficient as a semantic basis for sentences in Skolem normal form. Finally, we exhibit two proofs of an extension to Herbrand's theorem: every set of sentences of $L^{\mathsf{qB}^\bullet}$ in skolem normal form has a finite unsatisfiable set of ground instances.

11.1 Skolemization for qB

In a first-order calculus, skolemization is the process whereby prefix existential quantifiers of a sentence are eliminated in favor of new functions, called *skolem functions*, leaving only universal quantification. The resulting sentence is unsatisfiable if and only if the original one was. Skolem functions were first used by Skolem and later by Herbrand (see Kleene [28], page 343). Davis and Putnam [9] introduced the related concept of *clause form* for sentences of the predicate calculus, in which the only quantifiers are universal ones that appear at the beginning of a sentence, and the rest of the sentence is a disjunction of literals. In the next chapter we develop a clause form for **q**B; skolemization is a key part of that development.

In its original formulation, skolemization will not work with sentences of **q**B; that is, it will not preserve unsatisfiability. For example, consider the sentence $s =_{\mathrm{df}} \exists x.[S_i]Px \wedge \forall x.\neg[S_i]Px$. s certainly has no **q**B-models. However, eliminating the existential quantifier and replacing $\exists x.[S_i]Px$ by $[S_i]Px_0$ (where x_0 is a skolem constant) produces a sentence s' that is satisfiable. We can form a deduction structure that does not contain $P\hat{c}$ for any id constant $\hat{c}$ (satisfying $\forall x.\neg[S_i]Px$ by Definition 9.6), yet contains Px_0 because x_0 is not an id constant.

The solution to this is to maintain the special status of quantified-in variables by replacing them with bullet terms, just as was done in defining substitution for free variables in the definition of $\dot{L}^{\mathsf{qB}}$(9.2). Bullet terms in a modal context are schematic: they refer to a constant that is derived from the identifier functions $\boldsymbol{\eta}$.

Skolemization introduces the requirement that the language contain functional terms with arguments; to this point we have considered only constants. The extension of L^{qB} to include function terms is straightforward. However, we will also introduce a slightly different form for bullet terms. This form has the advantage of transparent substitution: any term can be substituted into the argument of the modal operator. Formerly, we had to convert the term into its bullet form and add an additional I-conjunct (see Definition 9.2). By having a complicated rule of substitution, we could keep the form of the language L^{qB} simple (*i.e.*, no bullet terms), which is useful when comparing it to other modal logics of belief. However, the nonuniform substitution process causes complications when used with the resolution rule of inference, and so we abandon it here. The price we pay is that the language becomes slightly more complicated, so that sentences in clause form contain bullet terms.

11.1.1 The Language $L^{\mathsf{qB}^\bullet}$

We first define the extended base language $L_0^\bullet$.

DEFINITION 11.1. *The language $L_0^\bullet$ is defined as in Definition 3.1, with the addition of the following items.*

6. *A denumerable set of* functions of degree n *for each* $n > 0$ *(generally small roman letters from the middle of the alphabet, e.g., f, g).*
7. *A term operator $\bullet$ of one argument.*

A term with no bullet operators is called an orthodox term. A formula of the form $\bullet(\tau)$, τ an orthodox term, is called a bullet term. *An orthodox term with no variables is called an* orthodox ground term. *Generally, the adjective "orthodox" will indicate that the object has no bullet terms, e.g.,* orthodox atom, orthodox sentences.

The standard formation and valuation rules for ordinary functional terms of a first-order language apply, for example those in Kleene [28], pages 148–150. The mapping φ from terms to individuals is extended to include functional terms in this manner. We define bullet terms for any orthodox term τ as $\bullet(\tau)$ (we will omit parentheses when τ is a simple term, *e.g.*, $\bullet a$). Note that this means that bullet terms are never nested, *e.g.*, $\bullet(f(\bullet a))$ is impermissible.

The language $L^{\mathsf{qB}^\bullet}$ is formed from the base language $L_0^\bullet$ in the usual manner, by the addition of modal operators (see Definition 9.1). However, we place some additional restrictions on what constitutes a *sentence* of $L^{\mathsf{qB}^\bullet}$. First, bullet terms must always be in the context of a modal operator, so that $\forall x.P(\bullet x, a)$ is forbidden. We do this so that we do not have to define the semantics of $\bullet$ as an orthodox function; it is strictly a schematic operator inside a modal context. Second, if $[S_i]\psi$ is a modal atom, all free variables of ψ (the quantified-in variables) must occur inside the scope of a bullet operator. Thus

$$\forall x.[S_i]\exists y.P(\bullet x, y)$$

and

$$\exists x.[S_i]P(\bullet(f(a, x)), a)$$

are sentences, but

$$\forall x.[S_i]\exists y.P(x, y)$$

and

$$\exists x.[S_i]P(\bullet(f(a, x)), x)$$

are not.

We now give the substitution rule for $L^{\mathsf{qB}^\bullet}$. Substitution into modal contexts is the same as in nonmodal contexts.

DEFINITION 11.2. *Let ψ be a formula of $L^{\mathsf{qB}^\bullet}$. For every variable x and orthodox term τ the formula ψ^x_τ is given inductively by the rules (1), (2), and (3) of Definition 3.2, together with the rule*

4. $([S_i]\psi)^x_\tau = [S_i]\psi^x_\tau$.

We carry over the valuation of a sentence of L^{qB} with respect to a model m from Definition 9.6. Note that, because of the condition that all quantified-in variables be under the scope of a bullet operator, there is never any necessity to check the truth of any I-predicate in m to determine the truth of a modal atom. For example, there is a difference in the meaning of $\exists x.[S_i]Px$ in L^{qB} and $\exists x.[S_i]P{\bullet}x$ in $L^{\mathrm{qB}^\bullet}$. The latter is true in a model that has an individual k such that $[\![Pk]\!]_{\eta_i} \in \mathrm{bel}(d_i)$; while the former is true only if $I_i(k)$ is true in addition. $\exists x.[S_i]Px$ is logically equivalent to $\exists x.I_i x \wedge [S_i]P{\bullet}x$ in $L^{\mathrm{qB}^\bullet}$. We can translate a sentence of L^{qB} into one of $L^{\mathrm{qB}^\bullet}$ using the following definition.

DEFINITION 11.3. *Let $[S_i]\psi'$ be an atom of $L^{\mathrm{qB}^\bullet}$ obtained from $[S_i]\psi$ by replacing every free variable x_j not under the scope of a bullet operator by ${\bullet}x_j$, and conjoining the atom $I_i x_j$. $[S_i]\psi'$ is called the* bullet transform *of $[S_i]\psi$.*

The truth value of a modal atom and its bullet transform are equivalent, as we now prove.

THEOREM 11.1. *Let $[S_i]\psi$ be a modal atom (either L^{qB} or $L^{\mathrm{qB}^\bullet}$) with the free variables* $\mathbf{x}$, *and $[S_i]\psi'$ its bullet transform. For any element sequence* $\mathbf{k}$, *$([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$ is logically equivalent to $([S_i]\psi')^{\mathbf{x}}_{\mathbf{k}}$.*

Proof. By inspection of the valuation rules of Definition 9.6, $([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$ is true in m just in case $(\psi^{\mathbf{x}}_{\mathbf{k}})_{\eta_i} \in \mathrm{bel}(d_i)$, and $I_i(k)$ is true in m for every member $k \in \mathbf{k}$ that is not under the scope of a bullet operator. These are seen to be precisely the truth conditions for $\psi'^{\mathbf{x}}_{\mathbf{k}}$ in m.∎

If we take a sentence s of L^{qB} and replace every modal atom of s by its bullet transform, then by Theorem 11.1 the resultant sentence s' is logically equivalent. Thus there is a translation of sentences from L^{qB} to $L^{\mathrm{qB}^\bullet}$. The reverse is not true, however, since there may be bullet terms in $L^{\mathrm{qB}^\bullet}$ such as ${\bullet}(f(x))$ that have no equivalent form in L^{qB}, which only has bullet constants. However, if we exclude such terms, the reverse translation also produces a logically equivalent sentence.

With the semantics of Definition 9.6, the substitution theorem 9.7 can be seen to hold for $L^{\mathrm{qB}^\bullet}$; the proof is an easy variant of the one given.

11.1.2 Skolem Normal Form

To transform a set of sentences into its skolem normal form, we first move all quantifiers not in the scope of a modal operator to the front of each sentence (prenex form), then remove all existential quantifiers by replacing them with suitable skolem functions and bullet terms. The resultant set of sentences, we prove, is unsatisfiable if and only if the original set is.

DEFINITION 11.4. *A set of sentences of* $L^{\mathsf{qB}^\bullet}$ *is in* prenex normal form *if it has the form* $Q_1x_1Q_2x_2\ldots Q_nx_n\ M$, *where the* Q_i *are quantifiers and* M, *the* matrix, *is a boolean combination of atoms.*

Quantifiers under the scope of a modal atom are allowed in prenex normal forms, *e.g.*, $\forall x.(Qx \wedge [S]\exists y.P(\bullet x, y)$. Just as in the first-order calculus, it is possible to find a logically equivalent prenex normal form for any sentence of $L^{\mathsf{qB}^\bullet}$. The rules in Robinson [59], pages 143-149, are an effective procedure for this purpose. The resulting sentence is logically equivalent to the original in any logic that has the standard interpretation of the quantifiers and boolean operators, *i.e.*, the quantification is over individuals in some domain, and the matrix consists of boolean combinations of predications over these variables.

As an example, we translate the sentence $\forall x.(\neg\exists y.(Rxy \wedge \forall z.[S_i]R(\bullet z, \bullet y)) \vee Px)$ into prenex form.

$$\begin{array}{l}
\forall x.(\neg\exists y.(Rxy \wedge \forall z.[S_i]R(\bullet z, \bullet y)) \vee Px) \\
\forall x.(\forall y.\neg(Rxy \wedge \forall z.[S_i]R(\bullet z, \bullet y)) \vee Px) \\
\forall x.(\forall y.\neg\forall z.(Rxy \wedge [S_i]R(\bullet z, \bullet y)) \vee Px) \\
\forall x.(\forall y.\exists z.\neg(Rxy \wedge [S_i]R(\bullet z, \bullet y)) \vee Px) \\
\forall x.\forall y.\exists z.(\neg(Rxy \wedge [S_i]R(\bullet z, \bullet y)) \vee Px)
\end{array}$$

The skolemization process eliminates existential quantifiers from prenex normal form sentences.

DEFINITION 11.5. *Let s be a sentence of $L^{\mathsf{qB}^\bullet}$ of the form $\forall \mathbf{x}.\exists y.\phi$. The* skolem reduced form *of s is the sentence $\forall \mathbf{x}.\phi^y_{g(\mathbf{x})}$, where g is a new function symbol of* arity *$|\mathbf{x}|$.*

By the definition of the language $L^{\mathsf{qB}^\bullet}$, the skolem function $g(\mathbf{x})$ will be under the scope of a bullet operator when it is substituted into the context of a modal atom. For example, the skolem reduced form of $\forall x.\exists y.(Pxy \wedge [S_i]P(\bullet x, \bullet y)$ is

$$\forall x.P(x, g(x)) \wedge [S_i]P(\bullet x, \bullet(g(x))).$$

DEFINITION 11.6. *The* skolem transform *W' of a set of sentences W of the language $L^{\mathsf{qB}^\bullet}$ is formed by putting each sentence of W into prenex normal form, and then successively taking the skolem reduced form of each sentence until no more prenex existential quantifiers remain. The introduced skolem functions must be new to the entire set of sentences. If $W = W'$, then W is said to be in* skolem normal form.

The key theorem of this subsection is that a set of sentences is satisfiable precisely when its skolem normal form is. To prove this, we show that each application of skolem reduction to a set of prenex sentences preserves satisfiability.

THEOREM 11.2. *A set of prenex sentences W of $L^{\mathsf{qB}^\bullet}$ is unsatisfiable if and only if its skolem transform is.*

Proof. Let $s \in W$ be a sentence of the form $\forall \mathbf{x}.\exists y.\phi$ (if none such exists, W is in skolem normal form). We wish to show that W has a model *iff* W' does, where W' has the sentence $s' = \forall \mathbf{x}.\phi^y_{g(\mathbf{x})}$ in place of s.

If direction: Let m be a model of W', so that $m \models s'$. Then for every element sequence $\mathbf{k}$, $m \models \phi^{\mathbf{x},y}_{\mathbf{k},g(\mathbf{x})}$. Now $g(\mathbf{x}) = k$ for some element k of m, and so, by the substitution theorem (9.7), it must be the case that $m \models \phi^{\mathbf{x},y}_{\mathbf{k},k}$. Hence $m \models s$, and any model of W' is also a model of W.

Only if direction: Let m be a model of W, so that $m \models s$. Then, for every element sequence $\mathbf{k}$, there is some element k such that $m \models \phi^{\mathbf{x},y}_{\mathbf{k},k}$. If g is a function not appearing in W, then we can construct a model

m' from m that satisfies all sentences of W, such that $g(\mathbf{k})$ is precisely the k for which $m \models \phi^{\mathbf{x},y}_{\mathbf{k},k}$ for every sequence $\mathbf{k}$. By the substitution theorem (9.7), it must be the case that $m' \models \phi^{\mathbf{x},y}_{\mathbf{k},g(\mathbf{k})}$ for every sequence $\mathbf{k}$. Hence m' is a model of W'.∎

We will need the following definitions of instances and ground instances for sentences in skolem normal form.

DEFINITION 11.7. *Let W be a set of sentences of L^{qB^*} in skolem normal form containing the sentence $s = \forall \mathbf{x}.\psi$. By an* instance of s *we mean a substitution $\psi^{\mathbf{x}}_{\boldsymbol{\tau}}$ of terms for the universal variables. A* ground instance of s *is an instance that has no free variables.*

11.2 Normative Herbrand Models

The intent of this subsection is to develop Herbrand models for L^{qB^*}. These models have the property that the domain of discourse is the set of orthodox ground terms of L^{qB^*}. Confining our attention to Herbrand models simplifies the task of searching the space of B-models with arbitrary domains. A further reduction in the space can be made by considering only those Herbrand models whose identifier functions $\boldsymbol{\eta}$ are fixed. These are called Herbrand models *normative in $\boldsymbol{\eta}$*. The key theorem of this subsection is that normative Herbrand models are a complete survey of the space of B-models, in the sense that, for any set W of sentences in skolem normal form, we can find $\boldsymbol{\eta}$ such that W is satisfiable if and only if it has a Herbrand model normative in $\boldsymbol{\eta}$.

We start with the definition of the Herbrand universe.

DEFINITION 11.8. *The* Herbrand universe *of the language L^{qB^*} is the set of all orthodox ground terms. A* Herbrand model *is a B-model whose domain of discourse U is the Herbrand universe of L^{qB^*}, such that $v_0(\tau) = \tau$ for all orthodox ground terms τ.*

A Herbrand model maps every orthodox ground term into itself. It is an important fact that these are the only models we actually need consider for the predicate calculus. That is, there is a mapping from first-order models with arbitrary domains to Herbrand models, such that any set of prenex sentences that is satisfied by the first-order model is satisfied by the Herbrand model. First-order models can thus be grouped into equivalence classes, each class being represented by a Herbrand model. This leads immediately to several important results, for example, the Skolem-Löwenheim theorem, which states that every satisfiable set of first-order sentences has a denumerable model.

The associated Herbrand model m' (with valuation function v_0') can be constructed from any member m of the equivalence class in the following way: the valuation $v_0'(P\tau)$ of any ordinary orthodox ground atom is determined by its truth-value in m, so that $v_0'(P\tau)$ *iff* $m \models P\tau$. Basically, any models that agree on their interpretation function v_0 are in the same equivalence class. The Herbrand model that is the representative of this class is completely determined by the valuation of its ground atoms, since its domain of discourse is the Herbrand universe, *i.e.*, the set of all orthodox ground terms.

For L^{qB^*}, there also exist equivalence classes of B-models with a Herbrand representative. In the Herbrand model derived from m, the deduction structures must be the same as in m, so that all closed belief atoms have the same truth value as in m. This works fine for closed belief atoms; but those that have quantified-in variables are still problematical. The important criterion in the construction of the associated Herbrand model is that the truthvalue of *every* atomic predication, including those with free variables, stay the same when the corresponding elements of the models m and m' are inserted for the variables. For example, we want the truthvalue of $(Px)^x_a$ in m' to be the same as $(Px)^x_k$ in m when the referent of a in m is k. Now a belief atom such as $([S_i]Px)^x_k$ is true in a model m if the sentence Pa is in $bel(d_i)$, where a is the constant given by $a = \eta_i(k)$. To preserve the truthvalue of this atom in the model m', we must construe the function η'_i to return a when given any term that denotes k in m. This motivates the following definition.

DEFINITION 11.9. *Let m be a* B*-model. The Herbrand model m' associated with m is defined by the following rules.*

1. *The domain of discourse of m' is the Herbrand universe.*
2. *The valuation $\varphi'(\tau)$ of any orthodox ground term τ is the term itself.*
3. *The valuation $v'_0(P\boldsymbol{\tau})$ of any ordinary orthodox ground atom is determined by its truthvalue in m: $v'_0(P\boldsymbol{\tau})$ iff $m \models P\boldsymbol{\tau}$.*
4. *The identifier function η'_i of any orthodox ground term τ is given by $\eta'_i(\tau) =_{\mathrm{df}} \eta_i(\varphi(\tau))$.*
5. *The deduction structures of m' are identical to those of m.*

This definition yields a unique associated Herbrand model for any given B-model. One of the important properties of the associated model is that it agrees with the original B-model on atomic predications.

LEMMA 11.3. *Let m' be the Herbrand model associated with a* B*-model m. Let $\mathbf{k}'$ be a finite sequence of orthodox ground terms, such that the referent of each of these terms in m is the corresponding member of the sequence $\mathbf{k}$. If ϕ is an atom with at most the free variables $\mathbf{x}$, then*

$$m \models \phi^{\mathbf{x}}_{\mathbf{k}} \quad \textit{iff} \quad m' \models \phi^{\mathbf{x}}_{\mathbf{k}'} .$$

Proof. There are two cases: ϕ modal or nonmodal. For the nonmodal case, let $\boldsymbol{\tau}' =_{\mathrm{df}} \boldsymbol{\tau}^{\mathbf{x}}_{\mathbf{k}'}$. The referents of $\boldsymbol{\tau}'$ in m are given by the sequence $\boldsymbol{\tau}^{\mathbf{x}}_{\mathbf{k}}$, since the referents of $\mathbf{k}$ are given by the sequence $\mathbf{k}'$. Then we have the following chain of equalities.

$$\begin{aligned} m' \models (P\boldsymbol{\tau})^{\mathbf{x}}_{\mathbf{k}'} \quad &= m' \models P\boldsymbol{\tau}' \\ &= m \models P\boldsymbol{\tau}' \quad \text{(by Definition 11.9)} \\ &= m \models (P\boldsymbol{\tau})^{\mathbf{x}}_{\mathbf{k}} \end{aligned}$$

For the modal case, we must show that $m' \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}'}$ *iff* $m \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$. Because m and m' have the same deduction structure d_i, this is true if $[\![\psi^{\mathbf{x}}_{\mathbf{k}'}]\!]^i_{m'} = [\![\psi^{\mathbf{x}}_{\mathbf{k}}]\!]^i_m$. By item (4) of Definition 11.9, $\eta'_i(k'_j) = \eta_i(k_j)$ for all j, and so these formulae are identical.∎

Finally, we are in a position to prove the key theorem about Herbrand models.

THEOREM 11.4. *Let W be a set of sentences of $L^{\mathsf{qB}^\bullet}$ in skolem normal form. W is unsatisfiable if and only if there is no Herbrand model of W.*

Proof. Let m be a model that satisfies W; we will show that its associated Herbrand model m' also satisfies W. Suppose there is a sentence $\forall\mathbf{x}.\psi$ of W that is falsified by m'. Then, for some sequence of orthodox ground terms $\mathbf{k}'$, we must have $m' \not\models \psi^{\mathbf{x}}_{\mathbf{k}'}$. Since the truth of ψ is a function of the truth of its constituent atoms, by the previous lemma it must be the case that $m \not\models \psi^{\mathbf{x}}_{\mathbf{k}}$, where $\mathbf{k}$ is the sequence consisting of the referents of $\mathbf{k}'$ in m. Hence $\forall\mathbf{x}.\psi$ is falsified by m, a contradiction.∎

As a consequence of this theorem, we can relate the unsatisfiability of a set of sentences of $L^{\mathsf{qB}^\bullet}$ in skolem normal form to the unsatisfiability of the set of ground instances of these sentences, where all quantifiers have been eliminated. We just note that if $s = \forall\mathbf{x}.\psi$ is falsified by some Herbrand model m, then, for some sequence $\mathbf{k}$ whose elements are in the Herbrand universe, $\psi^{\mathbf{x}}_{\mathbf{k}}$ is false in m. By the substitution theorem (9.7), this means that the *ground instance* $\psi^{\mathbf{x}}_{\boldsymbol{\tau}}$ of s must be false in m, where $\boldsymbol{\tau} = \mathbf{k}$. Hence if a set of sentences in skolem normal form is falsified by every model, the set of instances of these sentences is also.

COROLLARY 11.5. *The set of ground instances of an unsatisfiable set of sentences in skolem normal form is also unsatisfiable.*

This corollary is not quite Herbrand's theorem, because the set of ground instances is denumerably infinite when the Herbrand universe is. Using the compactness result for L^{qB} (Theorem 9.13), Herbrand's theorem for $L^{\mathsf{qB}^\bullet}$ follows. However, we will also prove Herbrand's theorem using the method of semantic trees, because it will be useful for deriving results about resolution procedures for $L^{\mathrm{qB}^\bullet}$. Before proceeding to this topic, we develop the concept of normative Herbrand models for $L^{\mathrm{qB}^\bullet}$.

Theorem 11.4 is important because it collapses the space of models with arbitrary universes to just models with a Herbrand universe. However, even this class of models is, in a sense, too large to be surveyed in attempting to show the unsatisfiability of a set of sentences. The basic problem is that the identifier functions $\boldsymbol{\eta}$ are unconstrained: in order to make sure that there is no model of a sentence, we must check every possibility for $\boldsymbol{\eta}$. This problem is similar to that of the unconstrained universe in arbitrary first-order models. Our solution there was to restrict our attention to just a single universe; in a like manner, we can restrict the $\boldsymbol{\eta}$-mappings of B-models to just a single one; models with a representative mapping of this sort are called *normative models.*

DEFINITION 11.10. *Let C_i be a denumerably infinite subset of schematic constants in the language $L^{\mathrm{qB}^\bullet}$, and let $\boldsymbol{\eta}_i$ be a one-one correspondence of the Herbrand universe of $L^{\mathrm{qB}^\bullet}$ to C_i for each agent S_i. A* B-*model that has $\boldsymbol{\eta} = \eta_1, \eta_2, \ldots$ as its identifier functions is called* normative in $\boldsymbol{\eta}$.

The sequence C_i of schematic constants play the same role for bullet terms that skolem functions do for existential quantifiers. That is, we fix beforehand the particular schematic constant that a bullet term is mapped to by η_i, just as, for existential quantifiers, we fix the skolem constant that denotes the existential individual. Also, the mapping for every different bullet term is a different schematic constant. Because the constants are *schematic*, they have no distinguishing characteristics relative to the deduction structure rules for an agent, and hence can participate in any deduction in which the particular identity of the id constant is not important.

THEOREM 11.6. *Any set of sentences of $L^{\mathrm{qB}^\bullet}$ that has a denumerable model has a normative model.*

Proof. Let W be a set of sentences satisfied by m with denumerable universe U. We will derive a model m' from m that is normative and satisfies W. The valuation function v_0' and domain of discourse U of m' are the same as in m. The $\boldsymbol{\eta}$ mappings of m' are derived as follows. Let C_i be a denumerable set of schematic constants none of which is in W, and let η_i be a one-one correspondence between U and C_i. The deduction structures of m' are defined as follows. Consider the belief atoms that appear in W. Let $[S_i]\psi$ be such an atom with at most the free variables $\mathbf{x}$. If $m \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$, then $[\![\psi^{\mathbf{x}}_{\mathbf{k}}]\!]^i_{m'}$ is in the base set of d_i'.

It is obvious that $m' \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$ if $m \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$ for every element sequence $\mathbf{k}$. The converse must also be true, for suppose $m' \models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$, but $m \not\models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$. Then there must be a proof of $[\![\psi^{\mathbf{x}}_{\mathbf{k}}]\!]^i_{m'}$ from the base sentences of d_i; any such proof would still be a proof if any other orthodox terms were substituted for the schematic constants C_i, by the nature of schematic constants. Thus there would also be a proof of $[\![\psi^{\mathbf{x}}_{\mathbf{k}}]\!]^i_m$ from the base sentences of d_i, which yields $m \not\models ([S_i]\psi)^{\mathbf{x}}_{\mathbf{k}}$, a contradiction.

Since the valuations of m and m' have the same domain of discourse and agree on every atomic valuation over that domain, they must agree everywhere. Hence m' is a model of W.∎

An important corollary of this theorem is that every set of sentences W that has a Herbrand model also has normative models; indeed, it has a model normative in $\boldsymbol{\eta}$ for *every* choice of the identifier ensemble $\boldsymbol{\eta}$, as long as the range of each function of $\boldsymbol{\eta}$ does not include any schematic constants that appear in W. This means that, for any W, we can fix $\boldsymbol{\eta}$ and just examine those Herbrand models normative in $\boldsymbol{\eta}$ when we try to show the unsatisfiability of W.

COROLLARY 11.7. *Let W be a set of sentences of $L^{\mathrm{qB}^\bullet}$, and let $\boldsymbol{\eta}$ be any identifier ensemble whose ranges contain no constants of W. Then if W has a model, it has one normative in $\boldsymbol{\eta}$.*

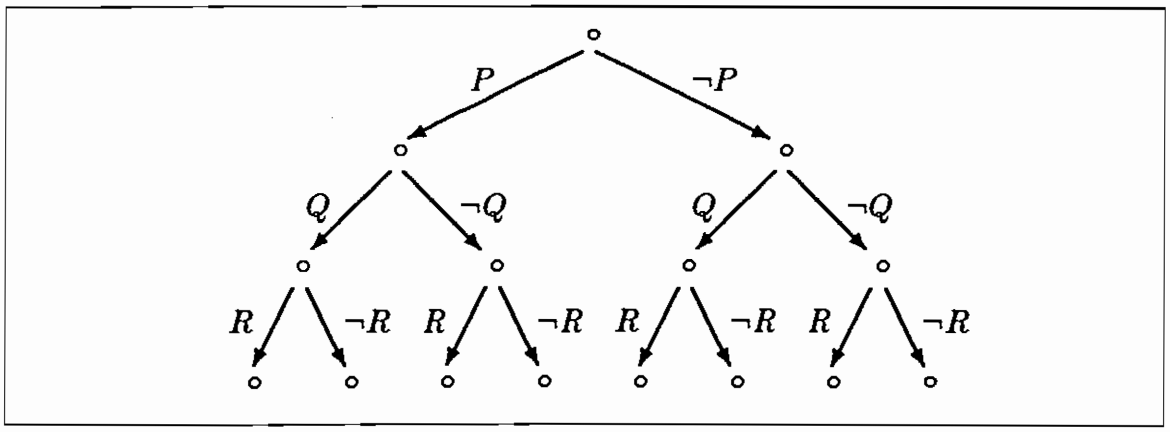

Figure 11.1 A Semantic Tree for P, Q, and R.

11.3 Semantic Trees

For first-order languages, the Herbrand models are completely surveyed by a type of unordered binary tree called a *semantic tree*. They are derived by considering a sequence P_1, P_2, ... of all ground atoms. Starting at the root node, the first element of the sequence is used to add two new nodes; the branch to one is labeled by P_1, to the other by its complement $\neg P_1$. The process is continued, so that, at a stage n, each of the leaf nodes is extended by adding two new nodes, and the branches are labeled by P_n and $\neg P_n$ (see Figure 11.1). Each branch of the tree, extended as far as possible (if the sequence is infinite, then the branch will be too) is a complete valuation of all ground atoms.

The *Herbrand base* is the set of all ground atoms of a first-order language. For a predicate calculus, a valuation of the Herbrand base defines a unique Herbrand model, since the Herbrand universe is the domain of discourse, and all function terms map to themselves. Hence a semantic tree surveys all possible first-order Herbrand models, with each complete branch being one such model.

For L^{qB^*}, the situation is more complicated. A B-model contains, in addition to its first-order component, a set of deduction structures representing the belief subsystems of agents, and a function ensemble $\boldsymbol{\eta}$ for associating constants with individuals. To survey these models in a semantic tree, we will put additional elements, namely ground orthodox modal atoms, into the Herbrand base. The effect of these atoms will be to encode the contents of deduction structures. Each branch of the semantic tree will then represent a set of Herbrand models: those with the same first-order and deduction structure components, but with different $\boldsymbol{\eta}$ functions.

DEFINITION **11.11.** *The* Herbrand base *of a language* $L^{\mathrm{qB}^\bullet}$ *is the set of ground orthodox atoms of* $L^{\mathrm{qB}^\bullet}$.

Recall that an atom is *ground* if it contains no free variables. For example, if $L_0 = \{P^2, Q^1, a, b, f^1\}$, then

$$\begin{array}{l} P(a,b) \\ Q(f(f(a))) \\ [S_1]P(a,b) \\ [S_2](Qa \wedge \forall x.P(x, f(a))) \end{array}$$

are all in the Herbrand base. Note that closed modal atoms can contain quantification and boolean operators under the scope of the modal operator, but do not have free variables or bullet terms. That is, for the Herbrand base, we are only interested in modal atoms asserting that a particular sentence is in a deduction structure, just as we are interested in ordinary atoms asserting a property for particular constants of the Herbrand universe.

DEFINITION **11.12.** *A* semantic tree *for a language* $L^{\mathrm{qB}^\bullet}$ *is a binary tree defined by considering a sequence* $\phi_1,\ \phi_2\ \ldots$ *of the Herbrand base of* $L^{\mathrm{qB}^\bullet}$. *Every node on the nth level of the tree has two daughters, and the links to these daughters are labeled with the literals* ϕ_n *and* $\neg\phi_n$. *The* literal set *of an arc is the set of literals on all links of the arc; on the arc ending in node N this is denoted by* $L(N)$. *A* branch *b is an arc that does not have any successors; it is* complete *if every member of the Herbrand base or its complement is in the literal set of b.*

An example of a finite semantic tree for $L^{\mathrm{qB}^\bullet}$ is given in Figure 11.2. As stated above, each complete branch of a semantic tree is intended to represent a set

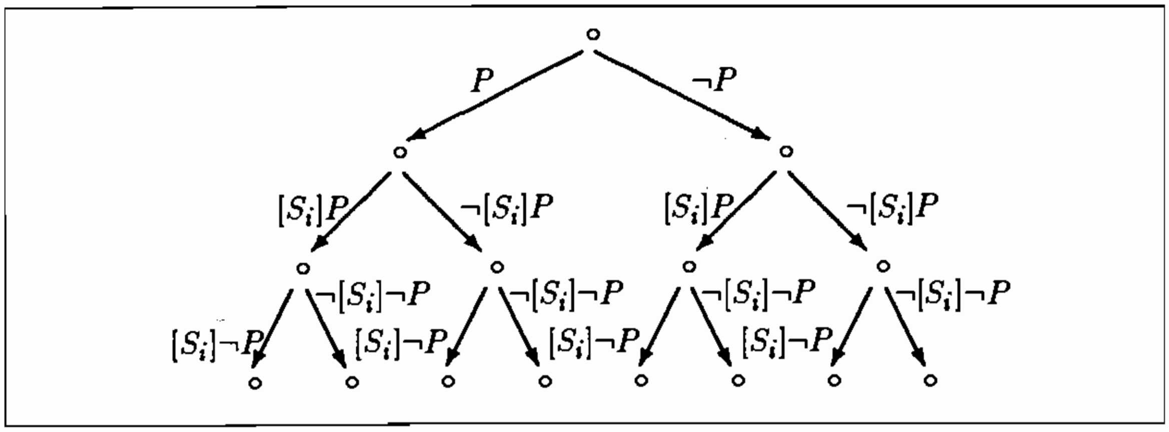

Figure 11.2 A Semantic Tree for P, $[S_i]P$, and $[S_i]\neg P$.

of Herbrand models that agree everywhere except on $\boldsymbol{\eta}$. If Z is the set of literals on a branch, then Z defines $v_0(P(\boldsymbol{\tau}))$ for all ordinary ground atoms by whether $P(\boldsymbol{\tau})$ or $\neg P(\boldsymbol{\tau})$ is in Z. It also defines deduction structures for each agent, by the condition that $p \in d_i$ if and only if $[S_i]p \in B$. However, there is an additional complication: not all branches define realizable deduction structures. For example, if $L(N) = \{[S_i]P, [S_i](P \supset Q), \neg[S_i]Q\}$, and $\rho(i)$ includes *modus ponens*, any complete branch that includes N does not define a deduction structure for S_i, because Q must be in a deduction structure that contains P and $P \supset Q$. Branches that contain such unsatisfiable sets of modal atoms are called *dismissed branches.*

Let us summarize this discussion of semantic trees with the following definition.

DEFINITION 11.13. *Let Z be the literal set of a complete branch b. An* associated Herbrand model of b *is any Herbrand model that satisfies the following conditions.*

1. $v_0(P\boldsymbol{\tau}) = \mathrm{t}$ *iff* $P\boldsymbol{\tau} \in Z$.
2. $p \in \mathrm{bel}(d_i)$ *iff* $[S_i]p \in Z$.

If b does not have any associated Herbrand models, then it is called a dismissed branch.

For the language $L^{\mathrm{qB}^\bullet}$, semantic trees are only an imperfect survey of the Herbrand models. Some complete branches, the dismissed branches, correspond to no Herbrand model at all, while the others have multiple associated models with different $\boldsymbol{\eta}$ ensembles. However, by Corollary 11.7, we know that, if a set of sentences has a Herbrand model, it has a normative one for any suitably defined ensemble. If we pick such an $\boldsymbol{\eta}$, then the semantic tree is a complete survey of all Herbrand models that are normative in $\boldsymbol{\eta}$: every complete, nondismissed branch specifies a first-order valuation v_0 and deduction structures d_i, while $\boldsymbol{\eta}$ and the Herbrand universe are fixed. From now on, we assume that we have picked a particular $\boldsymbol{\eta}$, and all Herbrand models of the tree are normative to this ensemble.

We now come to the key fact about semantic trees: if a set of sentences W in skolem normal form is unsatisfiable, we need only look at a finite frontier of the tree to show that this is so. A frontier node has the property that every complete branch passing through it leads to a normative Herbrand model in $\boldsymbol{\eta}$ that falsifies W. Since the tree is a complete survey of all such normative models (for fixed $\boldsymbol{\eta}$, of course), there is no satisfying Herbrand model, and hence no model at all.

We define what it means for a sentence to be *falsified* by the literal set $L(N)$ of a node. $L(N)$ can be thought of as only partially specifying a Herbrand model, or, equivalently, specifying a set of Herbrand models, the models that agree with its literal set (these models are all normative to a fixed $\boldsymbol{\eta}$). A sentence s is falsified by N if it is false in *every* Herbrand model normative to $\boldsymbol{\eta}$ that agrees with $L(N)$. If N is a maximal falsifying node, in the sense that none of its ancestors falsifies s, it is called a *failure node for* s, since any branch through it cannot lead to a Herbrand model satisfying s. A special type of failure node occurs when every continuation of an arc through a node N is a dismissed branch. N is called a *dismissed node*, and its valuation $L(N)$ does not agree with any Herbrand model; dismissed nodes thus falsify every sentence.

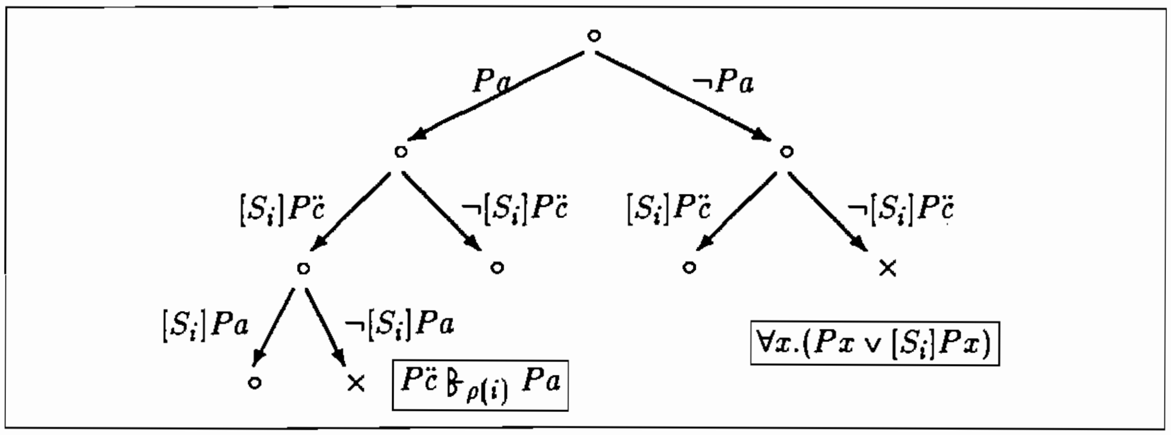

Figure 11.3 A Semantic Tree for P, $[S_i]P$, and $[S_i]\neg P$.

Example. Figure 11.3 shows a portion of the semantic tree for the sentence $s = \forall x.(Px \vee [S_i]Px)$ and the naming function $\eta_i(a) = \ddot{c}$. There are two failure nodes. The rightmost falsifies s, because its literal set includes $\neg Pa$ and $\neg[S_i]P\ddot{c}$: there is no Herbrand model normative in η_i that agrees with these literals and yet makes s true. The leftmost is a dismissed node, since $P\ddot{c} \Vdash_{\rho(i)} Pa$.

DEFINITION 11.14. *A semantic tree for s is* closed *if every branch* contains *a failure node for s.*

A closed semantic tree for W means that W has no Herbrand models normative in η, because the tree is a complete survey of all such models, and none satisfies W. We wish to show that W has a finite closed semantic tree whenever it has no Herbrand models normative in η. To prove this, we first establish a preliminary lemma about dismissed branches.

LEMMA 11.8. *Every dismissed branch has a failure node for tautologous s.*

Proof. A tautologous sentence s is true in every model, so we must show that every dismissed branch has a node that does not lead to *any* model. Let b be a dismissed branch, and suppose that, for every node N of b, the literal set $L(N)$ agrees with some Herbrand model. But then we can show that b must have an associated Herbrand model m, as follows. The first-order part v_0 given by Definition 11.13 is a valid atomic valuation, because no ordinary literal and its complement are present in b. Now consider the belief literals of b for agent S_i. This is a perhaps infinite set $\{[S_i]\Gamma, \neg[S_i]\Delta\}$ that contains no bullet terms. By the Q-Attachment Lemma 9.10, if there is no deduction structure that satisfies these literals, there must be a finite subset $\Gamma_0 \subseteq \Gamma$ and an element $\delta \in \Delta$ such that $\Gamma_0 \not\vdash_{\rho(i)} \delta$. However, there is a node N whose literal set $L(N)$ contains $[S_i]\Gamma_0$ and $\neg[S_i]\delta$, and this is a failure node, contradicting the original assumption. Thus the deduction structures defined by b are legitimate.∎

The proof of this lemma depends on the fact that belief literal sets of $L^{\mathsf{qB}^\bullet}$ are atomically compact: if the whole set is unsatisfiable, there must be a finite unsatisfiable subset. The dismissed branches of a semantic tree are precisely those that contain an unsatisfiable set of belief literals, and every dismissed branch will have a node N whose finite literal set is unsatisfiable.

We need the following result about finitary trees (*i.e.*, trees that always have a finite number of branches at each node).

LEMMA 11.9. (König's Lemma) *Every infinite finitary tree has an infinite branch.*

This lemma was proven by König [29] in 1926. We will exploit the contrapositive form: if a finitary tree contains no infinite branch, it must be finite.

Using these two lemmas, we now prove a version of Herbrand's theorem for $L^{\mathsf{qB}^\bullet}$.

THEOREM 11.10. *Every set of sentences of $L^{\mathsf{qB}^\bullet}$ in skolem normal form is unsatisfiable if and only if it has a finite closed semantic tree.*

Proof. If part: suppose there is a closed semantic tree for the set of sentences W. Then there is a failure node on every branch, and W has no Herbrand model normative in η. By Corollary 11.7, W has no Herbrand model at all, and by Theorem 11.4, W is unsatisfiable.

Only if part: suppose W is unsatisfiable and in skolem normal form. Assume that the semantic tree has a complete branch b with no failure nodes; by Lemma 11.8, this branch has an associated Herbrand model. But then m satisfies W, by the following argument.

Let $s = \forall \mathbf{x}.\psi$ be some sentence of W. If $m \not\models s$, then for some element sequence $\mathbf{k}$ of the Herbrand universe it must be the case that $m \not\models \psi^{\mathbf{x}}_{\mathbf{k}}$. But then there must be a node of b that falsifies $[\![\psi^{\mathbf{x}}_{\mathbf{k}}]\!]^{i}_{m}$, because the interpetation of $\psi^{\mathbf{x}}_{\mathbf{k}}$ is fixed by an interpretation of the finite number of ground atoms that constitute its matrix. This contradicts the original assumption that there were no failure nodes, and so s is satisfied by m. Because s was chosen arbitrarily, every sentence of W is satisfied.

Thus every branch is closed at some finite point. By König's Lemma there must be a finite closed semantic tree for W.∎

This is the version of Herbrand's theorem given in Chang and Lee [5]. The standard form of the theorem can be derived by considering the frontier of failure nodes.

THEOREM 11.11. *Every unsatisfiable set of sentences of $L^{\mathrm{qB}^{*}}$ in skolem normal form has a finite unsatisfiable set of ground instances.*

Proof. Consider the minimal frontier of failure nodes of the finite closed semantic tree for an unsatisfiable set W. Each of these frontier nodes falsifies a ground instance of a sentence of W. Since the set of failure nodes is finite (by König's Lemma), there is a finite set of ground instances of W that falsifies every node of the frontier, and hence (by Corollary 11.7) is unsatisfiable.∎

12. Resolution Methods for qB

We develop resolution methods for **qB** similar to those for first-order logic. These methods are founded on the extension to Herbrand's Theorem for the logic qB. Although they were originally developed for **qB**, because of the correspondence property they are also appropriate for modal logics with possible-world semantics that are equivalent to the saturated form of **qB**.

Resolution methods have a special importance for AI, because one of the principal methodologies of the field is to axiomatize a domain in some formal language, and then produce intelligent behavior by proving theorems about the domain. The main reasoning facility of many AI systems is an automatic theorem-proving program; one need only look at the popularity of PROLOG, a programming language for AI based explicitly on resolution theorem-proving methods, to see that this is so. The emphasis in AI is on incomplete methods that can easily be modified to prove theorems efficiently relevant to a particular application domain. Perhaps the dominant technique in this regard is *rule-based deduction* (see Nilsson [54]), which combines a certain restriction of the resolution principle called *unit resolution* with techniques for embedding control information in the axioms.

In this chapter we will develop a resolution-based deduction system for **qB**, then introduce refinements, including unit resolution, that lead to the definition of a rule-based deduction system. The basic resolution system, R, contains one resolution rule specifically dealing with belief operators. This rule is a version of the attachment rule A modified for clause form, and is noteworthy in two respects. First, it is similar to the *hyperresolution* rule defined by Robinson [60], in which input is taken from some number of clauses, rather than just two. Second, it uses a *schematic* belief derivation operator $\Pi \mathrel{\mathrm{B}\!\!\vdash}^{\theta}_{\rho(i)} \psi$ that allows its arguments Π and ψ to have free variables, and returns a substitution θ under which a derivation can be obtained. Together, these techniques alleviate the three computational inefficiencies of tableau systems pointed out at the beginning of the last chapter.

The system R is proved sound and complete. From it, we define three particularizations. The first, RK, uses resolution with answer extraction for schematic belief derivation, and is also a complete system. The second, RKu, is an incomplete but more efficient refinement of RK using only *unit resolution*: at least one resolvent is always a unit clause. Unit resolution is a technique employed by AI rule-based deduction systems, and the final method we define, based on RKu, is an incomplete system RB of this variety.

To the author's knowledge, this is the first resolution system developed for a modal logic based on a first-order language. The only other relevant work seems to be that of Fariñas-del-Cerro [12], who has developed a resolution technique for the predicate modal logic $S4$ with the Barcan formula as an axiom. However, there is a crucial restriction on the language he uses. Quantifying-in is allowed, so that an expression of the form $\exists x.\Box Px$ is legal; but quantifiers may not be introduced *within the scope of a modal atom*; so the expression $\Box(\exists x.Px)$ is not. This is a great simplification of the language, since it does away with any interaction between variables quantified inside and outside the scope of the modal operator. As we have seen in the development of Herbrand's Theorem in the previous chapter, it is precisely this interaction that necessitates the introduction of bullet terms, and causes complications in showing that Herbrand's Theorem actually holds. Fariñas' rules are not extendable to the language of **q**B.

12.1 Unification and Resolution

Resolution is a single inference rule that is sound and complete for sentences of the predicate calculus in clause form. In this section we derive resolution rules for the clause form of $L^{\mathsf{qB}^\bullet}$. Because of the added complexity of the language $L^{\mathsf{qB}^\bullet}$, two rules are necessary for a complete system.

12.1.1 Clause Form for $L^{\mathsf{qB}^\bullet}$ and L^{qB}

We wish to transform a set of sentences B of L^{qB} or $L^{\mathsf{qB}^\bullet}$ into a set of clauses of $L^{\mathsf{qB}^\bullet}$, which are simply disjunctions of literals prefixed by universal quantifiers. This can be done in four steps.

1. If the sentences are in L^{qB}, then replace each belief atom by its bullet transform.
2. Convert B to Skolem normal form.
3. Convert each matrix into conjunctive normal form.
4. Dissolve conjuncts to arrive at individual disjunctive clauses.

As an example, we translate the singleton set $\{\forall x.(\neg\exists y.(Rxy \wedge \forall z.[S_i]R(z,y)) \vee Px)\}$ into clause form. First, because this is a sentence of L^{qB}, we replace the belief atom by its bullet transform.

$$\forall x.(\neg\exists y.(Rxy \wedge \forall z.I_iz \wedge I_iy \wedge [S_i]R(\bullet z, \bullet y)) \vee Px) \quad .$$

Next, the prenex form of the sentence is

$$\forall x.\forall y.\exists z.(\neg(Rxy \wedge I_iz \wedge I_iy \wedge [S_i]R(\bullet z, \bullet y)) \vee Px) \quad .$$

Skolemizing yields

$$\forall x.\forall y.(\neg(Rxy \wedge I_ig(x,y) \wedge I_iy \wedge [S_i]R(\bullet(g(x,y)), \bullet y) \vee Px) \quad .$$

We convert the matrix into its conjunctive normal form

$$\forall x.\forall y.\neg Rxy \vee \neg I_i g(x,y) \vee \neg I_i y \vee \neg [S_i] R(\bullet(g(x,y)), \bullet y) \vee Px \quad .$$

There is only one clause, namely

$$\{\neg Rxy,\ \neg I_i g(x,y),\ \neg I_i y,\ \neg [S_i] R(\bullet(g(x,y)), \bullet y),\ Px\} \quad .$$

Note that a clause is a finite set of literals; it is equivalent in truthvalue to the disjunction of its members. At times we write clauses with disjunctions instead of set brackets, as in $L_1 \vee L_2 \ldots$. We use capital A to stand for a set of literals; the notation $C = L, A$ or $C = L \vee A$ indicates a clause C consisting of the set $\{L\} \cup A$. The null clause is written as $\blacksquare$.

Because each of the steps in translating to clause form preserves the satisfiability of the original set of sentences B, we have the following theorem:

THEOREM 12.1. *A set of sentences of $L^{\mathrm{qB}^\bullet}$ is unsatisfiable if and only if its clause form is.*

We will need the following definitions of instances and ground clauses.

DEFINITION 12.1. *Let C be a clause of $L^{\mathrm{qB}^\bullet}$ with at most free variables $\mathbf{x}$. By an* instance *of C we mean a substitution $C^{\mathbf{x}}_{\mathbf{t}}$ of terms for the universal variables. A* ground instance *of C is an instance that is a sentence.*

Because of the results of the previous section on normative Herbrand models, we have the following important theorem relating the satisfiability of a clause and its ground instances.

THEOREM 12.2. *A clause is unsatisfiable only if a finite subset of its ground instances is unsatisfiable.*

12.1.2 Substitution and Unification

We carry over results from Robinson [59] directly in this subsection. No modifications need be made for $L^{\mathsf{qB}^\bullet}$, because unification is insensitive to the semantics of the language it is applied to.

DEFINITION 12.2. *A* substitution *is a form* $\{\tau_1/x_1, \ldots \tau_n/x_n\}$, *where all of the* x_i *are unique and no* x_i *occurs in any* τ_j. *It is a* ground substitution *if each of* τ_j *is ground. If* θ *is a substitution* $\{\boldsymbol{\tau}/\mathbf{x}\}$ *and* C *is a clause,* $C\theta = C^{\mathbf{x}}_{\boldsymbol{\tau}}$ *is an instance of* C. *A substitution* θ *is* more general *than a substitution* σ *(*$\theta \succeq \sigma$*) if* σ *can be formed by the composition of* θ *and some other substitution.*

Unification is a matching operation in which two or more literals are made to look identical, if possible, by a given substitution. The substitution is called a *unifier* of the literals. If there is more than one unifier of a given set of literals, there will always be a *most general unifier* that has the property that it is more general than every other unifier.

DEFINITION 12.3. *A* unifier *of a set of literals is a substitution* θ *that makes the literals identical. If for every unifier* σ *of the literal set,* $\theta \succeq \sigma$, *then* θ *is a* most general unifier *or* mgu *of the set. We write* $L_1\theta L_2$ *if* θ *is the most general unifier of* L_1 and L_2.

Every unifiable literal set has a most general unifier, and all mgu's are equivalent modulo the renaming of variables. The *unification algorithm* (see Robinson [59], pages 191–192) is an effective procedure that takes a set of literals as input, and either terminates and reports that no unifier exists, or produces the most general unifier as output.

12.1.3 Complete Schematic Belief Derivation

In the ordinary predicate calculus, resolution is a rule of inference that derives a clause from two parent clauses. The general idea is to make a literal from one parent clause the complement of a literal in the other parent clause, by finding a most general unifier for their atoms. If $C_1 = L_1, A_1$ and $C_2 = L_2, A_2$ are the parent clauses, and L_1 and L_2 are complementary literals under a most general substitution θ, then the resultant clause is $R = A_1\theta, A_2\theta$. L_1 and L_2 are called the *literals resolved upon.*

Resolution is an interesting rule of inference for the predicate calculus because of the properties of the most general unifier. One important result of these properties is called the *Lifting Lemma* (see Robinson [59], pages 210–211): if $C_1' = L_1', A_1'$ and $C_2' = L_2', A_2'$ are any instances of C_1 and C_2, then their resolvent $R' = A_1'\theta', A_2'\theta'$ is an instance of R. Essentially, the Lifting Lemma says that when we do a resolution, we are considering at once a whole class of inferences that could have been performed on the instances of the parent clauses. A single resolution involving the most general unifier subsumes all these others.

In the case of modal literals, there are difficulties in applying a unification procedure. Consider first the simple case of the two literals $\neg[S_i]P{\bullet}x$ and $[S_i]P{\bullet}a$. Here the most general unifier is $\{a/x\}$, and the two modal atoms are identical under this substitution; a valid resolution inference can be made using the two complements.

On the other hand, consider the literals $\neg[S_i]P{\bullet}x$ and $[S_i](Qa \wedge P{\bullet}a)$. There is no unifier for the modal atoms, and hence no way to produce complementary literals by substitution. However, a valid inference does exist, given sufficiently powerful deduction rules $\rho(i)$, because $Qa \wedge P\hat{c} \Vdash_{\rho(i)} P\hat{c}$ for some constant $\hat{c}$ such that $\hat{c} = \eta_i(a)$. Thus there is a substitution $\{a/x\}$ that causes a conflict between the two literals, but it cannot be found by the application of the unification algorithm.

How do we go about finding substitutions that make belief literals conflict? If we limit ourselves to systems in which conflicting literals are determined by their attachment to the belief deduction operator, then the only possibility is to try every possible substitution that creates a ground instance, and test the resultant modal arguments using belief derivation. Needless to say, this is not an efficient procedure; it would be equivalent to doing resolution without unification, by creating ground instances of the parent clauses *first*, and then checking to see if they had literals that could be resolved upon.

To do better than this, we must allow the belief derivation process to return more information than simple success or failure. We have already addressed this point to some extent in Section 2.1, when we discussed the derivation of answers to questions of the form "What individuals have property P?" Such a question could be answered by a derivation process that allowed free variables in the query, and returned, along with an affirmative response, bindings for the free variables that made the query true.

In developing resolution rules for **qB**, we will assume that we have a derivation operator whose arguments can contain free variables, and whose result includes a substitution for those variables; we call this a *schematic belief derivation operator*. For example, we would write $Qa \wedge Pb \Vdash^{\{b/x\}}_{\rho(i)} Px$ if Pb were derivable from $Qa \wedge Pb$ under the rules $\rho(i)$. In general, the operator $\Vdash^{\theta}_{\mathcal{R}}$ describes a belief derivation process that uses the rules $\mathcal{R}$ and returns the substitution θ for free variables contained in its arguments. The substitution need not contain just ground terms; for example, the substitution $\{f(y)/x\}$ states that there is a proof for any individuals x and y that satisfy the given relationship.

It should be noted that there are cases in which no "most general" substitution exists for schematic belief derivation. For example, the derivation $Pa \wedge Pb \Vdash^{\theta} Px$ holds for either $\theta = \{a/x\}$ or $\theta = \{b/x\}$, and neither of these can be formed by composition from the other. However, it is important that the substitutions returned cover the space of possible instances of belief derivation. We make this more precise by stating the following condition on belief derivation:

Completeness Property for Schematic Belief Derivation. Let Γ and ψ be orthodox formulæ of $L^{\mathsf{qB}^{\bullet}}$, perhaps containing free variables. Let σ be an orthodox ground substitution such that $\Gamma\sigma \Vdash_{\rho(i)} \psi\sigma$. Then there is a schematic derivation $\Gamma \Vdash^{\theta}_{\rho(i)} \psi$, such that $\theta \succeq \sigma$.

A schematic derivation operator that has this property is said to be *complete*.

Do there exist derivation processes with the properties we have just described? In the simplest case, one could always take ordinary belief derivation and have it return substitutions by the following strategy. Assume there is some enumeration of ground substitutions $\theta_1, \theta_2, \ldots$. For Π and ψ with free variables, try to derive $\psi\theta$ from $\Pi\theta$. If it succeeds, then $\Pi \Vdash^{\theta}_{\rho(i)} \psi$, and the derivation process returns θ. Continuing in this manner, all of the ground substitutions that make the derivation succeed would eventually be found. If we are trying to show that clauses involving the modal atoms $[S_i]\Pi$ and $[S_i]\psi$ are unsatisfiable, by Theorem 12.2 there is a finite subset of instances of these clauses that are unsatisfiable, and at some point we would accumulate enough information from the belief derivations to show this.

Of course, this is just our original, inefficient suggestion for using belief derivation in resolution. But there are other methods that work better. One such is the *answer extraction* technique of Green [16]. We will give an example of this in a later section (12.2.1); here we give resolution rules for any complete schematic belief derivation process.

Besides being complete with respect to substitutions, we demand that belief derivation also be schematic with respect to a countably infinite set of terms. In the language L^{qB}, the only terms were constants, and we used a set of id constants called *schematic constants* (written as $\ddot{c}$). In schematic belief derivation we must allow for more complicated constructions, because the argument to a bullet operator can be a functional term, perhaps containing free variables. In performing belief derivations, we replace the bullet operators with a special class of unary function symbols, the *schematic functions*. These will be written as $\ddot{g}$, and have the same properties as schematic constants, namely, any derivation containing the schematic term $\ddot{g}(\tau)$ is a valid derivation if the schematic term is replaced everywhere by an id constant. Note that we do not demand that the argument to $\ddot{g}$ be schematic. In a sense, the schematic term isolates its argument from the rest of the sentence, in much the same way that the bullet operator does.

In an analogous fashion to Definition 9.3, we define the bullet deletion transform of a set of formulæ of $L^{\mathsf{qB}^\bullet}$. Note that these formulæ may have free variables.

DEFINITION 12.4. *Let Π be a set of formulæ of $L^{\mathsf{qB}^\bullet}$, and let $\ddot{g}$ be a unary schematic function not appearing in Π. The* bullet deletion transform of Π *is a set of sentences $\Pi^\bullet$ in which every bullet operator of Π is replaced by $\ddot{g}$.*

For example, consider the set of formulæ $\{P{\bullet}x, [S_i]P{\bullet}a\}$. Its bullet deletion transform is the set $\{P(\ddot{g}(x)), [S_i]P\ddot{g}(a)\}$.

The bullet deletion transform is closely related to the interpretation of the bullet operator under normative Herbrand models. We prove two theorems about this relationship.

THEOREM 12.3. *Let Γ and α be sentences of $L^{\mathsf{qB}^\bullet}$, and let m be a normative Herbrand model. Then $[\![\Gamma]\!]_m^i \Vdash_{\rho(i)} [\![\alpha]\!]_m^i$ if and only if $\Gamma^\bullet \Vdash_{\rho(i)} \alpha^\bullet$.*

Proof. Because m is a normative Herbrand model, any bullet term ${\bullet}\tau$ in Γ or α is replaced by the schematic constant $\eta_i(\tau)$, where $\eta_i(\tau) \neq \eta_i(\tau')$ for $\tau \neq \tau'$. Similarly, in the bullet deletion transform of Γ and α a bullet term ${\bullet}\tau$ is replaced by the schematic term $\ddot{g}(\tau)$. The schematic terms $\ddot{g}(\tau)$ and $\ddot{g}(\tau')$ act like schematic constants that are different exactly when $\tau \neq \tau'$. By the nature of schematic terms and constants, any one can substitute uniformly for another in belief derivation; hence the two derivations are equivalent. ∎

THEOREM 12.4. *Let $\{\Pi, \psi\}$ be a set of formulæ of $L^{\mathsf{qB}^\bullet}$, perhaps containing bullet terms and free variables, and let $\{\Pi^\bullet, \psi^\bullet\}$ be its bullet deletion transform. Suppose that $\Pi^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$. Then, for* *any ground substitution $\theta\sigma$ and any normative Herbrand model m, $[\![\Pi\theta\sigma]\!]_m^i \Vdash_{\rho(i)} [\![\psi\theta\sigma]\!]_m^i$.*

Proof. Suppose that $\Pi^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$. By the nature of complete schematic belief derivation, for any ground substitution $\theta\sigma$ we have $\Pi^\bullet\theta\sigma \Vdash_{\rho(i)} \psi^\bullet\theta\sigma$. By Theorem 12.3 above, this is equivalent to $[\![\Pi\theta\sigma]\!]_m^i \Vdash_{\rho(i)} [\![\psi\theta\sigma]\!]_m^i$ for any normative Herbrand model m.∎

We have abused the terminology slightly in the statement of these two theorems. The bullet deletion transform is an operation on a set of sentences considered as a unit; thus $\Pi^\bullet$ is the bullet deletion transform of Π, and $\psi^\bullet$ of ψ, but the set $\{\Pi^\bullet, \psi^\bullet\}$ is not the bullet deletion transform of the set $\{\Pi, \psi\}$. Nevertheless we will continue this notation when the context makes it clear what is meant, because it is more compact.

12.1.4 *The Resolution System* $\mathsf{R}(L_0, \rho)$

We now give three rules that form a complete resolution system for a given base language L_0 and agent rule ensemble ρ.

DEFINITION 12.5. *Let $\Vdash^{\theta}_{\rho(i)}$ be a complete schematic belief derivation operator. The resolution system $\mathsf{R}(L_0, \rho)$ has the following rules.*

$$R_1: \quad \frac{L_1, L_2, \ldots L_n, A}{L_1\theta, A\theta}, \quad \text{where } \theta \text{ is the most general unifier of } L_1,\ L_2,\ \ldots\ L_n.$$

$$R_2: \quad \frac{\begin{array}{c} L_1, A_1 \\ L_2, A_2 \end{array}}{A_1\theta, A_2\theta}, \quad \text{where } L_1\theta{\sim}L_2.$$

$$R_3: \quad \frac{\begin{array}{c} [S_i]\pi_1, A_1 \\ [S_i]\pi_2, A_2 \\ \vdots \\ [S_i]\pi_n, A_n \\ \neg[S_i]\psi, A \end{array}}{A\theta, A_1\theta, \ldots A_n\theta}, \quad \text{where } \pi_1^\bullet, \ldots \pi_n^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

Remarks. In all these rules, we assume that the variables in the parent clauses have been standardized apart: no two clauses share the same variable. The first rule is a *factoring* rule: it compacts literals of the same clause that can be made identical by some substitution. Rule R_2 is a binary resolution rule for predicate calculus. It finds literals of two parent clauses that can be made complementary by a substitution, and derives a new clause that does not contain the clashing literals. In many standard resolution systems for predicate calculus, the factoring and binary rules are combined into a single rule of resolution.

Rule R_3 is specific to belief literals. It is similar to the attachment rule qA for the sequent system qB, but it allows free variables in the premises, and belief derivation returns a substitution for these variables that is used in the conclusion. Note that the number of premises is variable: the only essential premise is the last one, the negative belief literal; all the others are optional.

Like the attachment rules for the sequent systems B and qB, rule R_3 uses the process of belief derivation directly in its definition. It is thus not a *bona fide* deduction rule unless schematic belief derivation is a decidable process. In the next section we define particular forms of belief derivation that allow us to consider systems in which the rules of belief derivation become rules of the resolution system itself. This is similar to the method used in deriving BK from B: instead of considering $\rho\!\!\vdash$ as a single operation, we break it down into steps, each of which becomes a rule of the logical system.

A *resolution proof* is defined in the following way. Let W_0 be a finite set of clauses whose unsatisfiability we wish to check (the *input clauses*). By choosing some parent clauses from W_0 and performing a resolution inference using one of the three rules above, we obtain a new clause C. Let $W_1 = W_0 \cup \{C\}$, and repeat the procedure for W_1. In this way we obtain a sequence of clause sets $W_0, W_1, \ldots$, such that any member of the sequence is unsatisfiable if and only if W_0 is. At some point, we may encounter a W_n that contains the null clause; if this occurs, then it must be the case that W_0 is unsatisfiable, and the sequence up to W_n counts as a proof of this fact.

Example. We first prove Theorem 9.1 of the system qB, namely, $\exists x.[S_i]Px \supset [S_i]\exists x.Px$. In clause form the negation of this sentence has three singleton clauses, with the skolem constant x_0:

$$\begin{array}{l} Ix_0 \\ [S_i]P{\bullet}x_0 \\ \neg[S_i]\exists x.Px \end{array}$$

If we assume that $P(\ddot{g}(x_0)) \Vdash_{\rho(i)} \exists x.Px$, then by rule R_3 the second two clauses can be resolved to produce the null clause. Note that we used the bullet deletion transform of $P{\bullet}x_0$, and so replaced ${\bullet}x_0$ by $\ddot{g}(x_0)$. Schematic belief derivation did not have to return any substitution, because there were no free variables.

For a slightly more complicated example, consider the following three clauses:

$$\begin{array}{l} [S_i]Q{\bullet}a \\ [S_i]P{\bullet}x, \neg[S_i]Q{\bullet}x \\ \neg[S_i](P{\bullet}a \wedge Q{\bullet}a) \end{array}$$

Assume that the rules $\rho(i)$ are strong enough to derive a conjunction from its two conjuncts. Then by using rule R_3 on all three clauses, and noting that $Q(\ddot{g}(a))$, $P(\ddot{g}(x)) \Vdash^{\{a/x\}}_{\rho(i)} P(\ddot{g}(a)) \wedge Q(\ddot{g}(a))$, we derive a fourth clause.

$$\begin{array}{l} [S_i]Q{\bullet}a \\ [S_i]P{\bullet}x, \neg[S_i]Q{\bullet}x \\ \neg[S_i](P{\bullet}a \wedge Q{\bullet}a) \\ \neg[S_i]Q{\bullet}a \end{array}$$

By either rule R_2 or R_3, the first and last clauses can be resolved to give the null clause.

12.1.5 Soundness and Completeness

The soundness of R_1 and R_2 with respect to first-order semantics is shown in Robinson [59]. To show that R_3 is sound, we first prove the following lemma about sets of belief atoms.

LEMMA 12.5. *Let $Z = \{[S_i]\Pi, \neg[S_i]\psi\}$ be a set of belief atoms of $L^{\mathsf{qB}^\bullet}$. If $\Pi^\bullet \nvdash^{\theta}_{\rho(i)} \psi^\bullet$, then every ground instance of $Z\theta$ is unsatisfiable.*

Proof. If $\Pi^\bullet \nvdash^{\theta}_{\rho(i)} \psi^\bullet$, then by Theorem 12.4, for every normative Herbrand model m and ground substitution $\theta\sigma$ we have $[\![\Pi\theta\sigma]\!]^i_m \nvdash_{\rho(i)} [\![\psi\theta\sigma]\!]^i_m$. Thus in m either one of $[\![\pi_i\theta\sigma]\!]^i_m$ isn't in bel(d_i), or $[\![\psi\theta\sigma]\!]^i_m$ is. Hence $Z\theta\sigma$ is unsatisfiable.

THEOREM 12.6. *The rule R_3 is sound.*

Proof. Let all parent clauses of R_3 be satisfied by some normative Herbrand model m. Let σ be an arbitrary substitution such that $\theta\sigma$ is a ground subsitution. The set of ground instances of the parent clauses under $\theta\sigma$ is also satisfied by m. By Lemma 12.5, one of the belief literals $([S_i]\pi_i)\theta\sigma$ or $(\neg[S_i]\psi)\theta\sigma$ must be false in m, and hence one of $A_i\theta\sigma$ or $A\theta\sigma$ must be true. Since this holds for an arbitrary ground substitution σ, every instance of the clause $A\theta, A_1\theta, \ldots A_n\theta$ must be true in m, and since m is a normative Herbrand model, the clause itself must be true in m.∎

The proof of completeness for first-order resolution has the following structure (see Chang and Lee [5]). Start by considering the unsatisfiable set of ground clauses that each unsatisfiable clause set gives rise to. This set has a finite closed semantic tree. The tree, if it is not empty, has at least one *inference node*: a node whose immediate successors are two failure nodes. The failed clauses are parent clauses for a resolution, the result of which is a clause that fails at the inference node or above. By doing successive resolutions of this sort, the closed frontier of the tree can be reduced to the root node, yielding the null clause. The Lifting

Lemma then applies, showing that a resolution proof must exist for the original set of clauses.

The proof of completeness of R follows this general line of argument. However, there are complications because failure nodes come in two flavors: those that falsify a clause because they contain the negation of literals in the clause, and those that falsify it because they contain a combination of belief literals that falsifies the belief literals in the clause. For example, consider a failure node N immediately dominated by the belief literal $[S_i]P$. Suppose this failure node falsifies a clause $C = \neg[S_i]P \vee A$, where none of the literals of A is falsified by $[S_i]P$. We call this failure node a *simple failure node* (or SFN) for C, and $\neg[S_i]P$ its *failing literal.* If, on the other hand, C is of the form $\neg[S_i]Q \vee A$, where $P \mathrel{\text{B}\!\!\vdash}_{\rho(i)} Q$, then the clause is falsified by the properties of belief derivation, and we call N a *complex failure node*, or CFN. CFN's do not have the inference property that SFN's do, because an inference node dominating a CFN does not always lead to an immediate resolution deduction that advances the frontier of the closed semantic tree. Most of the work in the completeness proof is concerned with showing that, even in the presence of complex failure nodes, there is always a *finite sequence* of resolution deductions with the inference property. We do this by first exhibiting a system R′ that recovers the inference property, and then showing that every deduction of the null clause in R′ is also a deduction in R.

First we introduce some terminology for failure nodes and their falsifying clauses. Let T be a semantic tree that enumerates the Herbrand models of $L^{\text{qB}^\bullet}$ for a fixed η. Recall from Section 11.3 that a falsifying node N for a clause C is such that every Herbrand model normative in η agreeing with $L(N)$ (all the literals on the arc from the root of the tree to N) falsifies C. A failure node is a maximal falsifying node, that is, there are no falsifying nodes between it and the root. The arc from the root to the failure node is called a *failure arc*; if N is a failure node, we will abuse the terminology slightly and call the arc from N to the root N also.

A concept that is useful is that of the *critical nodes* a belief literal with respect to a failure arc. Consider a failure arc N for a clause $C = L, A$. A critical node for L is one that, if we removed the portion of the arc between it and its parent, the arc would no longer falsify L. There may be several critical nodes if L is a belief literal, and N is a complex failure node for it. For example, suppose $P_1, P_2 \ldots P_n \vdash_{\rho(i)} Q$, and let $N = [S_i]P_1, [S_i]P_2, \ldots [S_i]P_n$ be a arc. N is a failure arc for the unit clause $\neg[S_i]Q$, and every node on N is a critical node.

If N is a dismissed node, it has no failing clause, but we can identify its critical nodes in the following manner. Because N is a dismissed node, there is some set of belief literals on its arc of the form $\{[S_i]P_1, [S_i]P_2, \ldots [S_i]P_n, \neg[S_i]Q\}$, where $P_1, P_2 \ldots P_n \vdash_{\rho(i)} Q$. Each of these literals immediately dominates a critical node of the arc.

We now develop the resolution system R' for ground clauses. This system uses a new notational device, the *amalgamated belief literal.*

DEFINITION 12.6. *An* amalgamated belief literal *is an expression of the form $[S_i]\{\Gamma; \alpha\}$. It is equivalent to the sentence $[S_i]\gamma_1 \wedge [S_i]\gamma_2 \wedge \ldots \wedge [S_i]\gamma_n \wedge \neg[S_i]\alpha$. Either Γ or α may be absent, but not both. If α is omitted, we write $[S_i]\{\Gamma\}$ without the semicolon. An amalgamated belief literal that is equivalent to $[S_i]p$ or $\neg[S_i]q$ is called* simple.

It should be stressed that amalgamated belief literals are merely a notational device, and do not introduce any new representational power into $L^{qB^\bullet}$. The system R' is defined for clauses that use the new notation. Thus, in the input clauses, every positive belief literal $[S_i]p$ has the form $[S_i]\{p\}$, and every negative belief literal $\neg[S_i]q$, $[S_i]\{;q\}$.

DEFINITION 12.7. *Let $\vdash_{\rho(i)}$ be a belief derivation operator. The resolution system $\mathsf{R}'(L_0, \rho)$ has the following rules.*

$$R_2': \quad \frac{\begin{array}{c} L, A_1 \\ \sim L, A_2 \end{array}}{A_1, A_2}, \quad \textit{where } L \textit{ is not a belief literal.}$$

$$\textit{Merge:} \quad \frac{\begin{array}{c} [S_i]\{\Gamma; \alpha\}, A_1 \\ [S_i]\{\Pi\}, A_2 \end{array}}{[S_i]\{\Gamma, \Pi; \alpha\}, A_1, A_2}, \quad \textit{where } \alpha \textit{ is optional.}$$

$$\textit{Reduce:} \quad \frac{[S_i]\{\Gamma; \alpha\}, A}{A}, \quad \textit{where } \Gamma^\bullet \Vdash_{\rho(i)} \alpha^\bullet$$

Remarks. These rules all involve ground literals only, so there is no need to introduce unification or schematic belief deduction. The soundness of R_2' and *Merge* follows from the rules of propositional logic; the soundness of *Reduce* is a simple consequence of Lemma 12.5. R' is also complete: if a set of ground clauses is unsatisfiable, there is a deduction of the null clause. We now prove this.

LEMMA 12.7. *Let N be an inference node whose daughters, M and M', are falsified by clauses C and C', respectively. There is a resolution deduction in R' from C and C' of a clause D that falsifies N.*

Proof. In what follows, we write $[\![\psi]\!]_{\eta_i}$ to indicate the interpretation of ψ as a belief sentence in any Herbrand model whose identifier function for S_i is η_i. We can do this because ψ contains no bullet terms, and the function η_i is sufficient to define the interpretation of ψ as a sentence of $L^{\mathsf{qB}^\bullet}$.

If both M and M' are simple failure nodes, then an application or R_2' to $C = L \vee A$ and $C' = \sim L \vee A'$ yields the resolvent $A \vee A'$, which falsifies N.

Suppose at least one of the failure nodes is complex. Let M be dominated by the belief literal $[S_i]q$, and M' by $\neg[S_i]q$. The general form of the clause C is $[S_i]\{\Gamma_j; \alpha_j\} \vee [S_i]\{\Pi_k\} \vee A$, and of C' it is $[S_i]\{\Gamma_m'; \alpha_m'\} \vee [S_i]\{\Pi_l'\} \vee A'$. All of the belief literals are failing literals for the respective nodes; both A and A', on the other hand, are falsified by N. Through a succession of applications of *Merge*, the clause $D = [S_i]\{\Gamma_j, \Pi_l'; \alpha_j\} \vee [S_i]\{\Gamma_m', \Pi_k; \alpha_m'\} \vee [S_i]\{\Pi_k, \Pi_l'\} \vee A \vee A'$ can be produced, where there is one amalgamated belief literal for every combination of indices j, l, k, l, and k, m. We must show that D falsifies N. We already know that A and A' are falsified at N. Consider the belief literal $[S_i]\{\Gamma_j; \alpha_j\}$. Let $[S_i]q, [S_i]\Lambda_j$ be its critical literals. Then by definition $\llbracket\Gamma_j\rrbracket_{\eta_i}, \Lambda_j, q \Vdash_{\rho(i)} [\alpha_j]_{\eta_i}$. Let $\neg[S_i]q, [S_i]\Lambda'_l$ be the critical literals of $[S_i]\{\Pi'_l\}$, so that $[\Pi'_l]_{\eta_i}, \Lambda'_l \Vdash_{\rho(i)} q$. By the transitivity of belief derivation, it must be that $[\Gamma_j]_{\eta_i}, \Lambda_j, \llbracket\Pi'_l\rrbracket_{\eta_i}, \Lambda'_l \Vdash_{\rho(i)} \llbracket\alpha_j\rrbracket_{\eta_i}$. Because the belief literals $[S_i]\Lambda_j$ and $[S_i]\Lambda'_l$ are falsified by N, so is the amalgamated belief literal $[S_i]\{\Gamma_j, \Pi'_l; \alpha_j\}$. A similar argument holds for $[S_i]\{\Gamma'_m, \Pi_k; \alpha'_m\}$ and $[S_i]\{\Pi_k, \Pi'_l\}$. Hence every disjunct of D is falsified by N.

Finally, we show that neither M nor M' can be a dismissed node. If they are both dismissed, then N must also be dismissed, because no branch of N leads to a valid normative Herbrand model; this contradicts the assumption that M and M' were failure nodes.

Assume that M' is a dismissed node dominated by $\neg[S_i]q$. M is falsified by some clause $C = [S_i]\{\Gamma_j; \alpha_j\} \vee [S_i]\{\Pi_k\} \vee A$, where A is falsified by N. Let $[S_i]q, [S_i]\Lambda_j$ be the critical literals of $[S_i]\{\Gamma_j; \alpha_j\}$; we must have $\llbracket\Gamma_j\rrbracket_{\eta_i}, \Lambda_j, q \Vdash_{\rho(i)} \llbracket\alpha_j\rrbracket_{\eta_i}$. Let $[S_i]\Lambda'$ be the critical literals of M', so that $\Lambda' \Vdash_{\rho(i)} q$. By the transitivity of belief derivation, it must be that $\llbracket\Gamma_j\rrbracket_{\eta_i}, \Lambda_j, \Lambda' \Vdash_{\rho(i)} [\alpha_j]_{\eta_i}$, and the amalgamated literal $[S_i]\{\Gamma_j; \alpha_j\}$ is falsified by N. Similar arguments show that the literals $[S_i]\{\Pi_k\}$ are also falsified by N. And, if we let M' be dominated by the positive belief literal $[S_i]q$, essentially similar arguments also show that any clause falsified by M is falsified by N. Hence N cannot be an inference node. ∎

This lemma shows that it is possible always to find a resolution deduction using R'_2 or *Merge* that advances the frontier of failure nodes. This leads directly to the following completeness theorem.

THEOREM 12.8. (Completeness of R') *If W is a set of ground clauses of L^{qB^*}, there is a derivation of the null clause in* R'.

Proof. By Lemma 12.7, there is always a sequence of resolutions that advances the frontier of failure nodes of the semantic tree. Thus there is some clause D that falsifies the root node, *i.e.*, every Herbrand model normative in $\boldsymbol{\eta}$. The only literals that are always false are amalgamated belief literals of the form $[S_i]\{\Gamma;\alpha\}$, where $[\![\Gamma]\!]_{\eta_i} \not\vdash_{\rho(i)} [\![\alpha]\!]_{\eta_i}$. By Theorem 12.3, it must be the case that $\Gamma^\bullet \not\vdash_{\rho(i)} \alpha^\bullet$, and the *Reduce* rule applies, yielding the null clause.∎

An important point to emerge from the proof of completeness is that there is always a deduction of the null clause from inconsistent W such that *Reduce* rules, if any, are all applied at the end of the deduction. We now seek a "normal form" for this deduction. It is first necessary to prove some lemmas on the commutativity of the order of R_2' and *Merge* applications.

LEMMA 12.9. *Let C_4 be derived from the input clauses C_1, C_2, and C_3 by first applying R_2' to C_1 and C_2, then applying Merge to C_3 and the result. Then there is a derivation of C_4 from the input clauses in which the Merge rule is applied after R_2'.*

Proof. Let $C_1 = [S_i]\{\Gamma;\alpha\} \vee A_1$, $C_2 = [S_i]\{\Pi\} \vee A_2$, and $C' = [S_i]\{\Gamma,\Pi;\alpha\} \vee A_1 \vee A_2$ the result of the *Merge* operation. Let $C_3 = L \vee A$, where L is the literal to be resolved on. $\sim L$ is in one or both of A_1 and A_2; let A_1' and A_2' be these sets with $\sim L$ deleted. Then $C_4 = [S_i]\{\Gamma,\Pi;\alpha\} \vee A_1' \vee A_2' \vee A$.

To generate C_4 starting with applications of R_2', consider first C_1. If it contains $\sim L$, then resolve it with C_3 to produce $C_1' = [S_i]\{\Gamma;\alpha\} \vee A_1' \vee A$; otherwise, let $C_1' = C_1$. Do the same for C_2, defining C_2'. Finally, merge C_1' and C_2', with the resolvent $[S_i]\{\Gamma,\Pi;\alpha\} \vee A_1' \vee A_2' \vee A = C_4$.∎

The order of successive *Merge*'s may also be commuted.

LEMMA 12.10. *Let C_4 be derived from the input clauses C_1, C_2, and C_3 by first applying Merge to C_1 and C_2, then applying Merge to C_3 and the result. There is a derivation of C_4 from the input clauses in which C_2 and C_3 are first merged, then the result is merged with C_1.*

Proof. Let $C_1 = L_1 \vee A_1$, $C_2 = L_2 \vee K_2 \vee A_2$, and $C_3 = L_3 \vee A_3$. Let $C' = L_{12} \vee K_2 \vee A_1 \vee A_2$ be the result of merging C_1 and C_2 using the literals L_1 and L_2, and $C_4 = L_{12} \vee L_{23} \vee A_1 \vee A_2 \vee A_3$ the result of merging C' and C_3 using literal K_2 and L_3. If we merge C_3 and C_2 first, we obtain the clause $C'' = L_{23} \vee L_2 \vee A_2 \vee A_3$; if we merge C'' with C_1 the result is $L_{23} \vee L_{12} \vee A_1 \vee A_2 \vee A_3 = C_4$.

If L_2 and K_2 are not distinct, the same proof will go through, except that, in the final clause C_4, there will be an amalgamated belief literal L_{123} composed of elements of L_1, L_2, and L_3.∎

We are now in a position to show that the system R can perform any derivation that R′ can. The way we do this is to indicate how a certain type of proof in R′, a *normal form* proof, can be converted to a proof in R. By "normal form" we mean a proof that has the following characteristics:

1. All R_2' deductions precede every *Merge* or *Reduce*.
2. All *Merges* precede all *Reduces*, and they produce a single clause $D = [S_i]\{\Gamma_j;\alpha_j\}$ containing only inconsistent amalgamated belief literals.
3. The tree of *Merges* leading to D can be reordered so that all merges that involve the literal $[S_i]\{\Gamma_1;\alpha_1\}$ are done after every other merge. There is thus a frontier of the merge tree $C_1 \ldots C_n$ such that $C_1 = [S_i]\{\Pi_1;\alpha_1\} \vee A_1$ and $C_j = [S_i]\{\Pi_j\} \vee A_j$ for $j > 1$, where $\Gamma_1 = \cup_{j=1}^{n}\Pi_j$, and every element of A_j is an inconsistent amalgamated belief literal.
4. We repeat the reordering process above for each of the clauses C_1 through C_n, choosing an inconsistent amalgamated belief literal from each. C_1 through C_n are *inconsistency nodes*, with the chosen literal called the *inconsistency literal*. The reordering process is continued recursively to the leaves of the tree, until the tree consists of subtrees dominated by inconsistency nodes.

A normal form proof is always obtainable for an unsatisfiable clause set W. Starting from the deduction tree obtained in the proof of Theorem 12.8, the *Merges* and R_2' applications are reordered (by Lemma 12.9) so that the latter always occur first. Then the merges are reordered (by Lemma 12.10) to conform to conditions 3 and 4.

THEOREM 12.11. *If the null clause can be deduced from the ground clauses W in* R′, *it can be deduced from W in* R *also.*

Proof. Let P' be a normal form proof in R$'$. We will convert P' into a proof P in R, since the proof P has essentially the same structure as P'.

Each application of R_2' in P' is converted into a parallel application of R_2 in P. To see that this is possible, note that the applications of R_2' use only clauses that have simple amalgamated belief literals. Because R_2' is an instance of R_2, these resolutions can be incorporated directly into P.

We have suceeded in converting the bottom part of P'; up to the frontier where merges begin, P' and P are identical. Because P' is in normal form, the merge part of the tree is segmented into subtrees dominated by inconsistency nodes. Choose a lowermost subtree, i.e., one whose leaves do not contain any inconsistency nodes. The frontier of this subtree consists of clauses of the form $C_1 = [S_i]\{;q\} \vee A_1$ and $C_j = [S_i]\{p_j\} \vee A_j$ for $j > 1$; the dominating node of the subtree is $C' = [S_i]\{p_2, p_3, \ldots; q\} \vee A_1 \vee A_2 \vee \ldots$. Because the first literal of C' is inconsistent, there is a deduction using R_3 of the clause $C = A_1 \vee A_2 \vee \ldots$ from the clauses C_1 and C_j. In P, we replace the complete subtree of P' dominated by C' with the clause C. Note that C is different from C' in that the inconsistent amalgamated belief literal is removed. Continuing in this manner, all the subtrees of P' can be replaced by instances of R_3. At the root of P will be the null clause, because the tree of merges is rooted in a clause consisting entirely of inconsistent amalgamated belief literals.∎

As a consequence of the last two theorems, R is provably complete on ground clauses.

COROLLARY 12.12. *If W is an unsatisfiable set of ground clauses of $L^{\mathsf{qB}^\bullet}$, there is a deduction of the null clause from W in* R.

Because every set of unsatisfiable clauses has a finite unsatisfiable subset of instances (Theorem 12.2), there always exists a ground deduction of the null clause from these instances. We now show that the Lifting Lemma holds: for every such ground deduction there is a deduction of the null clause from the original clause set.

LEMMA 12.13. *Let $C' =_{\mathrm{df}} L' \vee A'$ be a ground instance of the clause $C =_{\mathrm{df}} L \vee A$. Then there exists a ground substitution θ such that either $A\theta = A'$, or there is a factor $C'' = L'' \vee A''$ of C with $A''\theta = A'$.*

Proof. Let $C\sigma = C'$, since C' is an instance of C. If $A\sigma = A'$, the theorem holds; so assume $A\sigma \neq A'$. The only way this could happen is if there is some set of literals $L_1, L_2, \ldots L_n$ in A such that $L_i\sigma = L\sigma$. Then σ is a unifier of the literals $L, L_1, \ldots L_n$, and there is a most general unifier θ' of these literals. By the factoring rule R_1, there is a factor $C'' = C\theta'$ with the requisite property.∎

LEMMA 12.14. (Lifting Lemma) *Let C'_j be ground instances of the clauses C_j. If rule R_2 or R_3 derives C' from C'_j, then there is a deduction involving the same rule on the clauses C_j (or perhaps their factors) that derives C such that C' is a ground instance of C.*

Proof. For R_2, let $C_1 =_{\mathrm{df}} L_1, A_1$ and $C_2 =_{\mathrm{df}} C_2$ be two clauses, with $C'_1 =_{\mathrm{df}} L'_1, A'_1$ and $C'_2 =_{\mathrm{df}} L'_2, A'_2$ two of their ground instances. Suppose there is a deduction

$$R_2 \quad \begin{array}{c} L'_1, A'_1 \\ L'_2, A'_2 \\ \hline A'_1, A'_2 \end{array}$$

using the ground clauses as input. Define $C''_j =_{\mathrm{df}} L''_j, A''_j$ in the following way: if A'_j is an instance of A_j, then $C''_j = C_j$; else let C''_j be a factor of C_j such that A'_j is an instance of A''_j (by Lemma 12.13, this factor must exist). Now the literals L''_1 and $\sim L''_2$ must unify, because they have instances L'_1 and $\sim L'_2$, which must be identical for the foregoing deduction to occur; let $L''_j\sigma_j = L'_j$ be the relevant ground substitutions. We have assumed that all variables of the clauses do not conflict, so that, if we define $\sigma = \sigma_1\sigma_2$, we still have $L''_j\sigma = L'_j$. Let θ be the most general unifier of L''_1 and $\sim L''_2$; because $\theta \succeq \sigma$, there must be a substitution λ such that $\sigma = \theta\lambda$. Now there is a deduction

$$R_2 \quad \begin{array}{c} L''_1, A''_1 \\ L''_2, A''_2 \\ \hline A''_1\theta, A''_2\theta \end{array}$$

with $A_1''\theta\lambda, A_2''\theta\lambda = A_1', A_2'$.

For R_3, let $C_j =_{\mathrm{df}} [S_i]\pi_j, A_j$ and $C = \neg[S_i]\psi, A$ be a set of clauses, with $C_j' =_{\mathrm{df}} [S_i]\pi_j', A_j'$ and $C' = \neg[S_i]\psi', A'$ a set of their instances. Suppose there is a deduction

$$\frac{\begin{array}{c}[S_i]\pi_1', A_1' \\ [S_i]\pi_2', A_2' \\ \vdots \\ [S_i]\pi_n', A_n' \\ \neg[S_i]\psi', A'\end{array}}{A', A_1', \ldots A_n'}$$

with $\pi_1^\bullet, \ldots \pi_n^\bullet \Vdash_{\rho(i)} \psi^\bullet$. Again we define the clauses C'' and C_j'' as either the corresponding unprimed clause or its factor, so that $A''\sigma = A'$, and $A_j''\sigma_j = A_j'$. Let $\sigma' = \sigma\sigma_1\sigma_2\ldots\sigma_n$ be the composition of these subsitutions; since the variables of the clauses do not conflict, we still have $A''\sigma' = A'$ and $A_j''\sigma' = A_j'$. Because schematic belief derivation is complete, there is a substitution θ such that $\pi_1''^\bullet \ldots \pi_n''^\bullet \Vdash^\theta_{\rho(i)} \psi''^\bullet$, with $\theta \succeq \sigma'$. Hence there is a deduction

$$R_3\quad\frac{\begin{array}{c}[S_i]\pi_1'', A_1'' \\ [S_i]\pi_2'', A_2'' \\ \vdots \\ [S_i]\pi_n'', A_n'' \\ \neg[S_i]\psi'', A''\end{array}}{A''\theta, A_1''\theta, \ldots A_n''\theta}$$

with $A', A_1', \ldots A_n'$ an instance of the resolvent.▮

We now prove the completeness theorem for R.

THEOREM 12.15. *Let W be an unsatisfiable set of clauses of $L^{qB^\bullet}$. There is a deduction of the null clause from W using the rules of* R.

Proof. By Corollary 12.12, there is a deduction of the null clause using a finite set of ground instances of W. If P is a proof tree for the ground case, we can construct a proof tree P' using the clauses of W as follows. Starting at the leaves, we replace the ground instances by their dominating clauses. Then for every R_2 and R_3 deduction from the leaves, there is a corresponding deduction using the dominating clauses by the Lifting Lemma. The resulting resolvents have instances that are the corresponding resolvents in P. Continuing in this way, the tree P' can be generated, with each corresponding node in P being an instance of the node in P'. Finally, since the root of P is the empty clause, so is the root of P'.∎

12.2 Resolution Systems for Schematic Belief Deduction

In this section we define a form of R that characterizes the belief deduction process itself as a resolution system. Called RK, it uses resolution rules similar to those of Definition 12.5 for the agent's rules ρ. A key technique here is the use of *answer extraction* to make $\Vdash_{\rho(i)}$ a complete schematic belief derivation operator.

In this system, we make also make use of the *view indexing* technique that was part of B and qB. That is, since the outside observer and the agents are assumed to have similar proof methods, we simply index particular parts of the proof by a view to indicate what their status is: proving general facts about the actual world, or deriving beliefs in an agent's view.

12.2.1 Answer Extraction

Answer extraction is a technique whereby more information is recovered from a resolution proof than the simple conclusion that the input clauses are unsatisfiable. It was formulated originally by Green [16], and has the following motivation. Assume that there is some existential sentence $s = \exists \mathbf{x}.\psi$ whose validity we wish to show relative to a set of premises; and further, we wish to know for which individuals $\mathbf{x}$ the formula ψ is actually valid, *i.e.*, $\psi^{\mathbf{x}}_{\mathbf{\tau}}$ is valid. One possible method would be to insert particular terms for each of the free variables in ψ, and then try to prove the validity of the resultant instance. However, resolution methods afford a more efficient approach. If we negate s and put it into clause form, the resultant clauses will contain the universal variables $\mathbf{x}$. We now append these clauses to clauses produced by the premises, and attempt to show the whole set unsatisfiable. At each resolution step, we make note of the substitutions that are performed on the variables $\mathbf{x}$ (and if these are changed to new variables, we must keep track of that too). If the null clause is derived, then the substitutions for $\mathbf{x}$ will be the most general ones that could have been initially made, while still being able to complete the proof. Thus along with a proof comes a substitution for the originally existentially quantified variables.

In practice, this process of "answer extraction" is implemented by conjoining an *answer predicate* to the matrix of the existential sentence whose validity is in question. Thus the sentence s above would be changed to $s' = \exists\mathbf{x}.(\psi \wedge ANS(\mathbf{x}))$. In clause form, the negation of s' would have an ANS-literal in each of its clauses, *e.g.*, if $\psi = P(\mathbf{x}) \vee Q(\mathbf{x})$, then there would be the two clauses

$$\begin{array}{l} P(\mathbf{x}), ANS(\mathbf{x}) \\ Q(\mathbf{y}), ANS(\mathbf{y}) \quad . \end{array}$$

Note that the original variables have changed names in the second clause, and the answer predicate reflects this change.

In performing resolutions, the substitutions that are made for the original existential variables of s are always inserted into the answer predicate of a clause. If a singleton clause containing just this predicate is ever derived, then the resolution proof has succeeded, and the values of the answer pedicate's arguments are a substitution that makes the existential clause valid.

We will use the technique of answer extraction, but modify the original formulation slightly to take into account the demands of schematic belief derivation. Recall that the input to this derivation is a set of premise formulæ Π and a formula to be proven ψ; any of these formulæ may have free variables, although no two formulæ share the same free variable. First we form the variable sequence $\mathbf{x}$

consisting of just these free variables. We then put the set of formulæ $\{\Pi, \neg\psi\}$ into clause form, making certain to keep the original names of the free variables in all atoms. To each clause that contains such an atom, we add the atom $ANS(\mathbf{x})$. We now apply resolution rules as usual. If at any time the singleton clause $ANS(\boldsymbol{\tau})$ is deduced, then belief derivation has succeeded with the substitution $\{\boldsymbol{\tau}/\mathbf{x}\}$. We call the proof derived by this method an *answer-predicate analog* of the original proof.

Example. Suppose we are trying to perform the derivation $\forall y.(Py \supset Q(y,b))$, $P(\ddot{g}(x)) \Vdash^{\theta}_{\rho(i)} \exists y.Q(\ddot{g}(a),y)$. There is one free variable, x. The clause form is

$$\begin{array}{l} \neg Py, Q(y,b) \\ P(\ddot{g}(x)), ANS(x) \\ \neg Q(\ddot{g}(a),y) \quad . \end{array}$$

The only clause with an atom that originally contained a free variable is the second, and we add the literal $ANS(x)$ to this clause. Proceeding with resolution, we apply R_2 to the first and second clauses, producing

$$\begin{array}{l} \neg Py, Q(y,b) \\ P(\ddot{g}(x)), ANS(x) \\ \neg Q(\ddot{g}(a),y) \\ Q(\ddot{g}(x),b), ANS(x) \quad . \end{array}$$

This last clause can be resolved against the third, yielding the singleton clause $ANS(a)$. There is thus a proof of $\exists y.Q(\ddot{g}(a),y)$ from the premises, given the substitution $\{a/x\}$ for the original free variable x.

Although it was necessary to carry through the free variables unscathed in creating the original clause form, once the answer predicates have been added it is possible to change the variable names without affecting the outcome of the proof. It is the *position* of the variable within the answer predicate that encodes which original free variable is being substituted for.

The technique of answer extraction is applicable to any resolution-based deduction system. We use it in the following sections to convert ordinary belief derivation processes into schematic ones. The key theorem about answer extraction is that it is *complete*: every ground substitution that leads to a proof will be included in some substitution returned by the answer-extraction technique.

THEOREM 12.16. *Let Π be a set of clauses with schematic variables* $\mathbf{x}$, *and $\theta = \{\boldsymbol{\tau}/\mathbf{x}\}$ a ground substitution. Let $\mathcal{R}$ be a set of resolution rules for which the Lifting Lemma holds. If there is a resolution proof using $\mathcal{R}$ of the set $\Pi\theta$, then there is an answer-predicate analog that returns a substitution $\theta' \succeq \theta$.*

Proof. Suppose there is a derivation of the null clause from $\Pi\theta$. Because the Lifting Lemma holds, there is a derivation of the null clause from Π. If we append the answer predicate $ANS(\mathbf{x})$ to each clause of Π, this derivation will still be valid in its general form, but instead of deriving the null clause, it will derive a clause of the form $ANS(\boldsymbol{\tau}_1), ANS(\boldsymbol{\tau}_2), \ldots$. θ will be a unifier of these answer predicates, so a most general unifier θ' exists, with $\theta' \succeq \theta$.∎

The proof of this theorem depends on the Lifting Lemma holding for the resolution rules of belief derivation. It should be obvious from the above theorem that, by using the answer extraction technique, it is possible to turn simple belief derivation using resolution rules into *complete* schematic belief derivation.

12.2.2 *The Resolution Family* RK

In forming the family of systems $\mathsf{RK}(L_0, \sigma)$, we make the choice of a resolution-based refutation system for $\Vdash_{\sigma(i)}$. We also stipulate that the resolution rules σ, whatever their form, conform to the Lifting Lemma, so that they are amenable to answer extraction. In general, the rules σ will be modifications of the rules R_1–R_3 of the system R. For example, we can stipulate a *unit resolution system* or a *rule-based deduction system*, both of which are incomplete but efficient resolution methods (see Section 12.3 below).

To form the family RK, we define the derivation $\Pi \Vdash_{\sigma}^{\theta} \psi$ in precisely the way we did in the previous subsection, namely, as a refutation of the clause form of $\{\Pi, \neg\psi\}$ with answer extraction, using the rules σ. We also modify the rule R_3 of Definition 12.5 so that the individual steps of belief derivation become steps of a proof in RK. To do this, it is necessary to index these steps by the view, in a manner similar to that in BK. A resolution-based refutation proof begins with a finite set of input clauses, and consists of a series of inferences, each of which derives a new clause, until the null clause is derived (or, in the case of answer-analog proofs, until a singleton answer literal is derived). In performing a belief derivation within an ongoing refutation proof, we introduce a subsidiary structure of the same sort, called a *view window*, that contains its own set of input clauses and resolution inferences. Each view window is indexed by the particular view in which the belief derivation is taking place. In addition, it has the set of answer variables that are in force during the derivation, and an answer clause that will be returned if the proof succeeds. To sum up, we introduce the following definition.

DEFINITION 12.8. *A* view window ω *is a tuple* $\langle \nu, \mathbf{x}, C, W\rangle$. ν *is a* view, C *is a clause (the* answer clause*),* $\mathbf{x}$ *is a sequence of variables of* C *(the* answer variables*), and* W *is a set of clauses (the* input clauses*).*

There are four operations on windows that are appropriate for RK.

1. *Creating a window.* A window can be created based on a clause with a negative belief literal. For example, if $\neg[S_i]\psi, A$ is a clause, then a window for this clause would be $\langle i, \mathbf{x}, A, W\rangle$, where $\mathbf{x}$ are the variables of A, and W is the clause form of the bullet deletion transform of $\neg\psi$. If a refutation proof is ever found in this window, it constitutes the discharge of the negative belief literal, and the clause A can be returned as a valid deduction. This corresponds to the application of rule R_3 with a single premise.
2. *Adding positive belief literals to a window.* If $[S_i]\pi', A'$ is a clause containing a positive belief literal, then any window $\omega = \langle i, \mathbf{x}, A, W\rangle$ with view i can be enlarged by converting π' into the set of clauses W', and converting ω into $\omega' = \langle i, \{\mathbf{x}, \mathbf{x}'\}, \{A, A'\}, \{W, W'\}\rangle$, where $\mathbf{x}'$ are the variables of A'.

The ANS predicate would have to be modified every place it appeared in W, because if $\mathbf{x}'$ is nonempty, it will have more arguments.

This operation corresponds to application of the rule R_3 in which there are premises with positive belief literals.

3. *Rule application in a window.* The set of clauses W of a view window can be enlarged by application of one of the rules appropriate to that window.
4. *Returning an answer.* If the clause set W of a window $\omega = \langle i, \mathbf{x}, A, W\rangle$ contains a singleton answer literal $ANS(\boldsymbol{\tau})$, then the clause $A\{\boldsymbol{\tau}/\mathbf{x}\}$ can be returned as a deduction.

It should be clear that, in the case in which β is taken to be a resolution-based refutation system, the application of the four operations above generates the same inferences as the rule R_3. For suppose there is an instance of R_3 that derives the clause $A\theta, A_1\theta, \ldots$ from the premises $\neg[S_i]\psi, A,\ [S_i]\pi_1, A_1, \ldots.$ Then there must be a refutation proof that returns θ, using the bullet deletion transforms of the belief arguments. This proof can be reconstructed using operations on windows in the following manner. First the window is created using the clause $\neg[S_i]\psi, A$, and then expanded using each of the clauses $[S_i]\pi_j, A_j$. The rules $\sigma(i)$ are next applied as many times as necessary in the window to generate the proof, and finally the clause $A\theta, A_1\theta, \ldots$ is returned.

In the opposite direction, any time a result clause is returned from a window, there is an application of R_3 that would also produce the clause. The premises of R_3 are all of the clauses that went into the creation and expansion of the window; the derivation is the sequence of rule applications in the window. It should be noted here that derivation in windows is insensitive to the order of expansion of the window in the following sense: in any derivation of the answer predicate, there is a corresponding derivation in which all of the expansions take place before any of the rule applications.

The parallel between window operations and the alternate formulation of resolution using the *Merge* and *Reduce* rules of R' should be noted, although we have defined the latter only for ground clauses. When looked at in this manner, the clause $[S_i]\{\Gamma;\alpha\}, A$ is simply a view window whose input clauses are derived from $\Gamma^\bullet$ and $(\neg\alpha)^\bullet$, and whose answer clause is A. The *Merge* rule introduces more positive literals into a window, and also adds to the answer clause. The *Reduce* rule corresponds to a successful derivation in the view window, returning the answer clause as a result.

We can now give rules for the system $\mathsf{RK}(L_0,\sigma)$.

DEFINITION 12.9. *Let σ be a resolution rule ensemble. The system $\mathsf{RK}(L_0,\sigma)$ has the following rules.*

1. *The rules R_1 and R_2 of Definition 12.5.*
2. *The four rules for windows in Definition 12.8.*

Example. We take the rules $\sigma(i)$ to be the ordinary resolution rules R_1 and R_2 for each agent. Let B be the set of sentences (of L^{qB})

$$\begin{array}{l} \exists x.\neg[S_i]Qx \\ \forall xy.(\neg Rxy \supset [S_i]Px) \\ \forall x.[S_i](Px \supset Qx) \\ \forall x.\exists y.\neg Rxy \quad . \end{array} \tag{12.1}$$

In clause form (translating to $L^{\mathsf{qB}^\bullet}$) this yields the six clauses

$$\begin{array}{l} \neg I_i a, \neg[S_i]Q{\bullet}a \\ [S_i]P{\bullet}x, Rxy \\ I_i x, Rxy \\ [S_i](P{\bullet}x \supset Q{\bullet}x) \\ I_i x \\ \neg R(x, f(x)) \quad . \end{array} \tag{12.2}$$

Using the fifth clause, we can eliminate the predicate $I_i a$ from the first clause.

$$\begin{array}{l} \neg I_i a, \neg[S_i]Q{\bullet}a \\ [S_i]P{\bullet}x, Rxy \\ I_i x, Rxy \\ [S_i](P{\bullet}x \supset Q{\bullet}x) \\ I_i x \\ \neg R(x, f(x)) \\ \neg[S_i]Q{\bullet}a \end{array} \tag{12.3}$$

The last clause can be used to start a window with view i. We will write these windows as a set of clauses preceded by a line indicating the view ν, answer variables $xbar$, and return clause C, in the form view $i : \mathbf{x}, C$. In this instance, there are no answer variables, and the clause to return is empty:

$$\begin{array}{l} \text{view } i : \emptyset, \emptyset \\ \quad \neg Q(\ddot{g}(a)) \end{array}$$

We now add the belief literal from the fourth clause of (12.3) to the window.

$$\begin{array}{l} \text{view } i : \emptyset, \emptyset \\ \quad \neg Q(\ddot{g}(a)) \\ \quad \neg P(\ddot{g}(z)), Q(\ddot{g}(z)) \end{array}$$

It is important that the same function $\ddot{g}$ was used in converting the bullet terms for the added clause. There is still no return clause or answer variables.

There is a resolution that can be performed in the window, and that is done next.

$$\begin{array}{l} \text{view } i : \emptyset, \emptyset \\ \quad \neg Q(\ddot{g}(a)) \\ \quad \neg P(\ddot{g}(z)), Q(\ddot{g}(z)) \\ \quad \neg P(\ddot{g}(a)) \end{array}$$

Using the belief literal of the second clause of (12.3), we add another clause to the window.

$$\begin{array}{l} \text{view } i : x, Rxy \\ \quad \neg Q(\ddot{g}(a)) \\ \quad \neg P(\ddot{g}(z)), Q(\ddot{g}(z)) \\ \quad \neg P(\ddot{g}(a)) \\ \quad P(\ddot{g}(x)), ANS(x) \end{array}$$

The singleton clause Rxy has been added as the return clause of the window, and the answer variable is x. A final resolution step yields a singleton ANS clause.

$$\begin{array}{l} \text{view } i : x, Rxy \\ \quad \neg Q(\ddot{g}(a)) \\ \quad \neg P(\ddot{g}(z)), Q(\ddot{g}(z)) \\ \quad \neg P(\ddot{g}(a)) \\ \quad P(\ddot{g}(x)), ANS(x) \\ \quad ANS(a) \end{array} \tag{12.4}$$

At this point, the window proof has succeeded with the substitution $\{a/x\}$, and so the clause Ray is returned to the original set of clauses in (12.2).

$$\begin{array}{l} \neg I_i a, \neg[S_i]Q\bullet a \\ [S_i]P\bullet x, Rxy \\ I_i x, Rxy \\ [S_i](P\bullet x \supset Q\bullet x) \\ I_i x \\ \neg R(x, f(x)) \\ \neg[S_i]Q\bullet a \\ Ray \end{array}$$

The sixth and the last clauses can be resolved to yield the null clause (the substitution is $\{a/x, f(a)/y\}$).

12.3 Unit Resolution and Rule-based Deduction Systems

Unit resolution is a refinement of resolution in which we consider only those resolvents that have a unit parent clause (Chang and Lee [5], p. 133). Unit resolution is an efficient strategy, because it always produces resolvents with fewer literals than its largest parent. However, it is an incomplete rule: there are some valid sentences that cannot be proven by unit resolution. Rule-based deduction systems can be viewed as a refinement of unit resolution in which control information about which resolutions are to be performed is encoded by the syntactic form of the axioms. We give an example of a simple system of this type.

12.3.1 Unit Resolution for RK

Unit resolution can be incorporated into $\mathcal{R}$ by defining an appropriate refinement of R_3, and achieves a particularly simple and efficient form of this rule. The basic idea is to make all but one of the parent clauses unitary. There are two cases to consider:

$$R_3' : \quad \frac{\begin{array}{c}[S_i]\pi_1 \\ [S_i]\pi_2 \\ \vdots \\ [S_i]\pi_n \\ \neg[S_i]\psi, A\end{array}}{A\theta}, \quad \text{where } \pi_1^\bullet, \ldots, \pi_n^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$R_3'' : \quad \frac{\begin{array}{c}[S_i]\pi_1, A \\ [S_i]\pi_2 \\ \vdots \\ [S_i]\pi_n \\ \neg[S_i]\psi\end{array}}{A\theta}, \quad \text{where } \pi_1^\bullet, \ldots, \pi_n^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

In both these rules there can be any number of positive belief atoms. These atoms essentially comprise a base set of beliefs that can always be used in the belief derivation process that is incorporated into the resolution rule. Given these belief atoms $[S_i]\Gamma$ as background information, the two foregoing rules reduce to the following simple form:

$$R_3' : \quad \frac{\neg[S_i]\psi, A}{A\theta} , \quad \text{where } \Gamma^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$R_3'' : \quad \frac{\begin{array}{c}[S_i]\pi, A \\ \neg[S_i]\psi\end{array}}{A\theta} , \quad \text{where } \Gamma^\bullet, \pi^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

As with unit resolution for first-order languages, the rules R_3' and R_3'' always return a resolvent that is shorter than the largest of their parent clauses.

We can modify the window rules of RK to reflect the unit resolution refinement of $\mathcal{R}$. We do this in two parts: the rules of RK that create and expand windows are changed so that only unit resolutions of R_3 are performed; and rule application within a window is restricted to unit resolution, thus interpreting $\Vdash$ as unit resolution. The resulting system is called RKu. Its window rules are as follows:

1a. *Rule R_3'.* A window is created from $\neg[S_i]\psi, A$ in the normal way.

1b. *Rule R_3''.* A window is created using the unit clause $\neg[S_i]\psi$ and the clause $[S_i]\pi, A$. The answer clause is A, and the input clauses of the window are formed from the bullet deletion transform of $\pi \wedge \neg\psi$.

2. This is the same as the original, with the restriction that only unit clauses $[S_i]\pi'$ may be used.

3. The resolution rules applied in the window are unit resolutions.

4. Same as the original.

Remarks. There are two creation operations, corresponding to the two forms of the unit resolution rule for belief literals. In the second window operation, the addition of positive belief literals is restricted to unit clauses. With these modifications, it can be easily shown that the window operations generate the same inferences as the unit resolution rules R'_3 and R''_3, when $\beta\!\!\vdash$ is assumed to be a resolution-based refutation system. Note that, because of the restriction in the second window operation to unit clauses, the answer clause will always be shorter than the longest input clause used to create the window.

Example. We will use the same input clauses (12.3) as in the previous example.

$$\begin{array}{l}
\neg I_i a, \neg[S_i]Q{\bullet}a \\
[S_i]P{\bullet}x, Rxy \\
I_i x, Rxy \\
[S_i](P{\bullet}x \supset Q{\bullet}x) \\
I_i x \\
\neg R(x, f(x)) \\
\neg[S_i]Q{\bullet}a
\end{array}$$

The second and last clauses can be used to start a window with view i, according to (1b). The clause to return is the unit clause Rxy, and the answer variable is x. We also add the third clause, which is a single positive belief literal, as a background belief.

$$\begin{array}{l}
\text{view } i : x, Rxy \\
\quad \neg Q(\ddot{g}(a)) \\
\quad P(\ddot{g}(x)), ANS(x) \\
\quad \neg P(\ddot{g}(y)), Q(\ddot{g}(y))
\end{array}$$

At this point we perform two unit resolutions within the window, and derive a singleton answer clause.

$$\begin{array}{l}
\text{view } i : x, Rxy \\
\quad \neg Q(\ddot{g}(a)) \\
\quad P(\ddot{g}(x)), ANS(x) \\
\quad \neg P(\ddot{g}(y)), Q(\ddot{g}(y)) \\
\quad \neg P(\ddot{g}(a)) \\
\quad ANS(a)
\end{array}$$

This is the same window as (12.4), and the proof is concluded as in the previous example.

12.3.2 Rule-Based Deduction Systems

The currently favored technology for deduction in AI systems is *rule-based* (Nilsson [54], Chapter 6). Rule-based systems are efficient, easy to understand, and allow the incorporation of domain-specific knowledge to guide deductions. Their primary characteristics are:

1. Sentences of the system are divided into two classes: *facts* about the world, and *goals* to be proven.
2. Sentences need not be in clause form, although they usually have prenex universal quantifiers and are skolemized.
3. Facts whose main connective is material implication are treated as *rules*: forward-chaining rules that derive new facts from old ones, or backward-chaining rules that derive new subgoals from old ones.
4. *Matching* of goals and facts to rules is a key operation; it makes the application of rules similar to unit resolution.

Because of the distinction between facts and goals, rule-based systems can be viewed as having the proof structure of block tableaux. In the sequent $\Gamma \Rightarrow \Delta$, we can interpret Γ as being a set of facts, and Δ a set of goals. A proof in a rule-based system is simply a closed tableau, in which the sequent to be proven is the root of the tableau. However, the tableau rules that are employed are different from those we have seen in the system $\mathcal{T}_0$, and are based on the notion of matching (unification) and the use of implications as rules.

To illustrate these ideas, and to show how to accommodate belief derivation in this paradigm, we present a simple rule-based system. All sentences are assumed to be in prenex normal form, with existential quantifiers skolemized. We use a reverse implication sign "$\subset$"; it has the meaning $p \subset q \;\equiv\; q \supset p$.

Axiom. $\quad \Gamma \Rightarrow \blacksquare, \Delta$

MA. $$\frac{L_1, \Gamma \Rightarrow L_2 \wedge A, \Delta}{L_1, \Gamma \Rightarrow L_2 \wedge A, A\theta, \Delta}, \quad \text{if } L_1 \theta L_2$$

FC: $$\frac{L_1, L_2 \supset A, \Gamma \Rightarrow \Delta}{L_1, L_2 \supset A, A\theta, \Gamma \Rightarrow \Delta}, \quad \text{if } L_1 \theta L_2$$

BC: $$\frac{L_1 \subset A, \Gamma \Rightarrow L_2, \Delta}{L_1 \subset A, \Gamma \Rightarrow L_2, A\theta, \Delta}, \quad \text{if } L_1 \theta L_2$$

Remarks. All these rules use the basic matching operation of unification: $L_1 \theta L_2$ is true when θ is the most general unifier of the literals L_1 and L_2. The single axiom schema gives the closure condition for tableaux—the null clause appears as a goal. The matching rule *MA* is used to break down conjunctive goals and to derive the null clause from singleton goals. Forward chaining is accomplished by the rule *FC*, which generates new facts using forward implications. *BC* does backward chaining; note that the way in which the implication is written (normal or reversed implication sign) makes the difference in whether it is used to forward or backward chain.

The forward and backward chaining rules are simple variations of unit resolution, given that an implication $L \supset A$ is equivalent to the disjunction $\sim L \vee A$. Chaining rules differ from unit resolution in two ways. First, A need not be a disjunct of literals, but the unit resolution principle holds just the same: $\sim L \vee A$ and L' will derive the sentence $A\theta$, if θ is the most general unifier of L and L'. Second, chaining rules give more control over which literals are unified, and what happens to the resultant resolvent. Moore [50] has indicated how this control can be used to govern the pattern of deductions made, and make domain-specific inferences more efficient. Note that the control resides in the way the axioms are written for the domain, not in the tableau rules *FC* and *BC*. For this reason, rule-based systems have been called *procedural deduction systems.*

The rules just presented are relatively simple in that they do not specify what to do with more complicated types of goals, *e.g.*, conjunctive goals that do not match any fact literal. The way in which different rule-based systems handle these complexities is often an interesting and important part of these systems, and is another characteristic that distinguishes them from standard resolution refutation procedures. In the case of conjunctive goals, *splitting* is a popular technique: two sequents are generated, each of which must be proven in order to prove the original sequent (note the similarity to the conjunction rule C_2 of $\mathcal{T}_0$). However, to a large extent these concerns are independent of the problem we address here, which is how to accommodate belief literals in forward and backward chaining rules. For illustration purposes the foregoing rules are adequate.

We note one modification of the rules that will be useful. Answer extraction can be accomplished by conjoining a phantom *ANS*-predicate to the initial goal. This predicate is never "seen" by the axioms or rules, but every time a substitution occurs, the answer variables are updated. If the tableau closes, the *ANS*-predicate will have the requisite substitution for the variables of the input sentences of the proof.

Now we would like to write rules like *FC* and *BC* that use belief literals in the matching process. As we have seen in developing R, the appropriate matching operation is not unification, but schematic belief derivation. By noting the close resemblance between rule-based deduction and unit resolution, we can use the two rules R_3' and R_3'' to derive two forward-chaining and two backward-chaining rules.

$$MAb. \qquad \frac{[S_i]\Gamma, \Sigma \Rightarrow [S_i]\psi \wedge A, \neg[S_i]\Delta, \Lambda}{[S_i]\Gamma, \Sigma \Rightarrow A\theta, [S_i]\psi \wedge A, \neg[S_i]\Delta, \Lambda}, \quad \text{if } \Gamma^\bullet, \Delta^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$FCb_1: \qquad \frac{[S_i]\psi \supset A, [S_i]\Gamma, \Sigma \Rightarrow \neg[S_i]\Delta, \Lambda}{[S_i]\psi \supset A, [S_i]\Gamma, A\theta, \Sigma \Rightarrow \neg[S_i]\Delta, \Lambda}, \quad \text{if } \Gamma^\bullet, \Delta^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$FCb_2: \qquad \frac{\neg[S_i]\psi, \neg[S_i]\pi \supset A, [S_i]\Gamma, \Sigma \Rightarrow \neg[S_i]\Delta, \Lambda}{\neg[S_i]\psi, \neg[S_i]\pi \supset A, [S_i]\Gamma, A\theta, \Sigma \Rightarrow \neg[S_i]\Delta, \Lambda}, \quad \text{if } \pi^\bullet, \Gamma^\bullet, \Delta^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$BCb_1: \quad \frac{[S_i]\pi \supset A, [S_i]\Gamma, \Sigma \Rightarrow [S_i]\psi, \neg[S_i]\Delta, \Lambda}{[S_i]\pi \supset A, [S_i]\Gamma, \Sigma \Rightarrow [S_i]\psi, A\theta, \neg[S_i]\Delta, \Lambda}, \quad \text{if } \pi^\bullet, \Gamma^\bullet, \Delta^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

$$BCb_2: \quad \frac{\neg[S_i]\psi \supset A, [S_i]\Gamma, \Sigma \Rightarrow \neg[S_i]\Delta, \Lambda}{\neg[S_i]\psi \supset A, [S_i]\Gamma, \Sigma \Rightarrow A\theta, \neg[S_i]\Delta, \Lambda}, \quad \text{if } \Gamma^\bullet, \Delta^\bullet \Vdash^{\theta}_{\rho(i)} \psi^\bullet$$

Remarks. The matching rule *MAb* for belief literals is a straightforward use of the resolution rule R_3, where the clauses are all singleton belief literals. The first two sequent rules implement forward chaining. By converting $[S_i]\psi \supset A$ to $\neg[S_i]\psi \vee A$, the correspondence to the unit resolution rule R_3' is easily seen, where every parent is a unit clause except $\neg[S_i]\psi \vee A$. Note the presence of the background facts $[S_i]\Gamma$ and background goals $\neg[S_i]\Delta$. The second forward rule is similar, but comes from the unit resolution rule R_3'', in which the nonunit clause contains a positive belief literal. The implementation of backward-chaining is done in an analogous manner. Note the complete symmetry here: for each independent choice of positive or negative belief literal to match, and forward or backward direction, there is one rule.

We now define the system RB, a rule-based system for the language of qB. In RB, we take the belief derivation operator $\Vdash$ itself to be a rule-based deduction system with *ANS*-predicate capability. Note that this type of system gives use a large amount of flexibility in specifying deduction strategies for agents, even if the tableau rules are fixed, because most of the control information resides in the implicational facts agents have.

DEFINITION 12.10. *The system* RB *is a block tableau system with the following elements.*

1. *The null sentence axiom and rules MA, FC, BC, MAb, FCb_1, FCb_2, BCb_1, and BCb_2.*
2. *Belief derivation in each of the rules FCb_1, FCb_2, BCb_1, and BCb_2 is rule-based deduction, where the rules $\rho(i)$ of each agent are the rules (1).*

Example. We again use the same example, starting with the sentences (12.1). Because they are already in prenex form, we can skolemize all existential variables, and eliminate the leading universal variables. We let the first sentence be the goal, and the rest be facts.

Facts	Goals
$[S_i]P{\bullet}x \subset \neg Rxy$	$I_i a \wedge [S_i]Q{\bullet}a$
$I_i x \subset \neg Rxy$	
$[S_i](Q{\bullet}x \subset P{\bullet}x)$	
$I_i x$	
$\neg R(z, f(z))$	

These sentences are in almost the same form as the originals. We have chosen to use backward chaining for the implications, and so have used the reverse implication sign. Note that we have split the second sentence of 12.1 into two backward-chaining rules, instead of having a conjunction as the matching part of the reverse implication.

Lists of facts and goals are a nice alternative notation for sequents (these are the *analytic tableau* of Beth in [63]), and we use it now in preference to the latter. Whenever we apply a sequent rule of RB, we simply add to the lists. The first rule to apply is the matching rule MA, reducing one of the conjuncts of the goal:

Facts	Goals
$[S_i]P{\bullet}x \subset \neg Rxy$	$I_i a \wedge [S_i]Q{\bullet}a$
$I_i x \subset \neg Rxy$	$[S_i]Q{\bullet}a$
$[S_i](Q{\bullet}x \subset P{\bullet}x)$	
$I_i x$	
$\neg R(z, f(z))$	

We can apply the backward-chaining rule BCb_1, using the second goal and the first two facts. To do this, we open a view window to show that the relevant belief derivation goes through.

view $i : x, \neg Rxy$

Facts	Goals
$P\ddot{g}(x) \vee ANS(x)$	$Q\ddot{g}(a)$
$Q\ddot{g}(y) \subset P\ddot{g}(y)$	

The backward-chaining rule BC applies here, and we generate a new goal.

view $i : x, \neg Rxy$

Facts	Goals
$P\ddot{g}(x) \vee ANS(x)$	$Q\ddot{g}(a)$
$Q\ddot{g}(y) \subset P\ddot{g}(y)$	$P\ddot{a}$

By the matching rule MA, the tableau closes with answer predicate $ANS(a)$, and so we return from the window with $\neg Ray$. The rule BCb_1 has been successfully applied, and the original list of goals is augmented by the result.

Facts	Goals
$[S_i]P{\bullet}x \subset \neg Rxy$	$I_i a \wedge [S_i]Q{\bullet}a$
$I_i x \subset \neg Rxy$	$[S_i]Q{\bullet}a$
$[S_i](Q{\bullet}x \subset P{\bullet}x)$	$\neg Ray$
$I_i x$	
$\neg R(z, f(z))$	

Finally, the matching rule MA applies, and this tableau also closes.

13. Conclusion

13.1 Summary

We have explored a formalization of a computational paradigm of belief called the deduction model. It is interesting that the methodology we used was to examine the cognitive structure of AI planning systems. This methodology, which we might term *robot psychology*, offers some distinct advantages over its human counterpart. Because the abstract design of such systems is open and available, it is possible to identify major cognitive structures, such as the belief subsystem, that influence behavior. Moreover, these structures are likely to be of the simplest sort necessary to accomplish some task, without the synergistic complexity so frequently encountered in studies of human intelligence. The design of a robot's belief subsystem is based on the minimum of assumptions necessary to ensure its ability to reason about its environment in a productive manner: namely, it incorporates a set of logical formulas about the world, and a theorem-proving process for deriving consequences. The deduction model is derived directly from these assumptions.

The deduction model fits within that finely bounded region between formally tractable but oversimplified models, and more realistic but less easily axiomatized views. On the one hand, it is a generalization of the formal possible-world model that does not make the assumption of consequential closure, and so embodies the notion that reasoning about one's beliefs is resource-limited. On the other hand, it possesses a concise axiomatization in which an agent's belief deduction process is incorporated in a direct manner, rather than simulated indirectly. Perhaps the most fruitful idea in the thesis is to take the computational nature of agents' reasoning seriously, to such an extent that it becomes an integral part of the logics describing belief. In the tableau methods of Chapters 3 and 9, the belief derivation operator is used as part of the attchment rule that deduces consequences of statements concerning belief. If we have such an operator in hand, for example the particular proof process used by an agent reasoning about a specific domain, then we can simply "plug it in" to the deduction logic to produce a formal system that axiomatizes the agent's beliefs. Thus the deduction model and its associated logics B and qB thus lend themselves to implementation in mechanical theorem-proving processes as a means of giving AI systems the capability of reasoning about beliefs.

The resolution proof methods that were developed in the second half of the thesis promise to be an important practical advance in implementing this goal. Because of the correspondence property, these methods can also be used for possible-world models, and perhaps make existing systems like KAMP (Appelt [1]) more efficient.

The correspondence property is an interesting technical result of the thesis. Given a suitable common language for talking about the deduction and possible-world models, it turns out that the same statements are true of each of these models, as long as the deduction model is restricted to having sound and complete belief derivation for agents. So, despite the negative results of Montague, we are able to reconcile two divergent approaches to belief: the syntactic, symbol-processing paradigm whose origins are with Frege, and the more recent model-theoretic approaches of Hintikka and Kripke. We are free to interpret the language in whatever way satisfies our intuitions about belief.

13.2 Future Directions

As in most thesis endeavors, the hard part was knowing when to stop. There are many more topics that need to be covered before a truly coherent theory of deductive belief can be soundly established; this thesis formulates only a portion of that theory.

Two of the major issues we treated only partially were practical proof methods and theoretical computational issues. Regarding the former: although we developed resolution and rule-based systems for commonsense reasoning about belief, we have almost no experience in the practical application of these techniques in AI systems. They show promise of efficiency, but have yet to be tested; and although we do not actually expect it, it is conceivable that computational problems could limit their applicability. This is one of the most urgent areas for future work.

A second topic that was mentioned throughout the thesis but not really developed was to employ the deduction model to answer questions of a theoretical nature about belief as computation. We came closest to this in the theory of introspection, and already have some preliminary results that are not included in the thesis, of the following nature. Suppose we are given a robot agent with a belief subsystem that always returns an answer in a finite amount of time. Our goal is to give the agent some knowledge of his own beliefs, that is, to convert his belief subsystem into an introspective one. What is the best possible introspective behavior we can hope to achieve, in terms of the properties of faithfulness and fulfillment defined in Chapter 7, while still retaining the finite character of the computation? The answer turns out to be total faithfulness. That is, the attempt to establish either positive or negative fulfillment may result in altering the finite character of the belief derivation process for the agent.

On a more ambitious scale, we can consider trying to analyze the computational properties of other extended inference rules by using the deduction model. Our ideas here are admittedly tentative and vague; but the framework of viewing nonmonotonic inference as deduction over a complete theory seems to be a fruitful one. The first attempt here might be to develop further the theory of ignorance presented in Section 5.2. It may be possible to answer questions about computational limitations on reasoning about ignorance, along the lines suggested for introspective reasoning.

Finally, the deduction model of belief must be integrated with other cognitive processes if we are to develop truly intelligent robot agents that can perform commonsense reasoning. We have mentioned in the introduction some of the problems that were not addressed: belief revision and the interaction of belief and other components of cognition being the most important. This is a task that has barely begun.

Glossary

Languages

L_0	first-order language with constant terms only.
L	agents' language. Usually assumed to be the same as the system's language.
L^{B}	language of the systems $\mathsf{B}(L_0, \rho)$. Has modal operators $[S_i]p$, where p is a sentence (no quantifying-in).
L^{qB}	language of the systems $\mathsf{qB}(L_0, \rho)$. Has modal operators $[S_i]\phi$, where ϕ is a formula that may have free variables (quantifying-in).
$L^{\mathsf{qB}^\bullet}$	language of the resolution system $\mathcal{R}$. Has bullet terms of the form $\bullet\tau$.

Tableau Systems

$\mathcal{T}_0$	First-order complete rules for block tableaux (Definition 3.4).
$\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$	$\mathcal{T}_1$ is a subsystem of $\mathcal{T}_2$ (Definition 3.5). If $T_1 \sqsubseteq \mathcal{T}_2$ and $T_2 \sqsubseteq \mathcal{T}_1$, then $T_1 \simeq T_2$.
$Q_1 \rightarrow Q_2$	tableau system Q_1 extends Q_2 (Definition 7.2).

Deduction Structures

$d = \langle B, \mathcal{R} \rangle$	deduction structure with base set B and rules $\mathcal{R}$ (Section 2.2).
d_ν	deduction structure of the view ν.

$\mathsf{D}(L, \mathcal{R})$	class of deduction structures formed from a language L and rules $\mathcal{R}$.

Models

$\mathsf{B}(L, \rho)$-model $=< v_0, \varphi, \mathsf{U}, \{\ldots d_i \ldots\} >$	class of models parameterized by a language L and rule sets $\rho(i)$ for every agent S_i. v_0, φ, and U give a first-order valuation of L, and the deduction structures d_i give an interpretation of the modal atoms of L (Definition 4.4).
B^+-model	Model for the situation logic B^+ (Section 5.4). Consists of a sequence of B-models.

Logic Families

$B(L, \rho)$	basic belief logic family. L is the agents' language, ρ an ensemble of deduction rule sets, one for each agent (Definition 3.7).
BK	nonintrospective belief logic family, using the attachment rule A_K(Definition 3.9).
BK_s	saturated BK, in which the rules of BK are admissible for every agent.
BSn	introspective belief logic families $\mathsf{BS4}$ and $\mathsf{BS5}$ (Definition 7.1).

Expressions

Modal	An atom of the form $[S_i]\phi$.
Circumscriptive	An atom of the form $\langle S : \Gamma\rangle p$. Means that p follows from Γ in S's belief subsystem (Section 5.2).
Ordinary	A nonmodal atom or sentence.
Ground	A formula with no free variables.
Orthodox	A formula with no bullet terms.

Appendix A. Proof of the Wise Man Puzzle

We solve the Wise Man Puzzle using the decision procedure *DPP* developed in Section 10.3 for saturated B^+. **s1**, **s2**, and **s3** are the three wise men. **t1** is the initial situation, **t2** the situation after the first wise man speaks, and **t3** after the second speaks. **p1** is the proposition that the first wise man's spot is white, **p2** the second's, and **p3** the third's. The axioms are:

```
axiom W1 p1∧p2∧p3
```

In the real world, everyone's spot is white.

```
axiom W2 [F,t1](p1∨p2∨p3)
```

It's a common belief that at least one spot is white.

```
axiom W3 [F,t1](p1⊃[s2,t1]p1)
axiom W3 [F,t1](p1⊃[s3,t1]p1)
axiom W3 [F,t1](p2⊃[s1,t1]p2)
axiom W3 [F,t1](p2⊃[s3,t1]p2)
axiom W3 [F,t1](p3⊃[s1,t1]p3)
axiom W3 [F,t1](p3⊃[s2,t1]p3)
axiom W3 [F,t1](¬p1⊃[s2,t1]¬p1)
axiom W3 [F,t1](¬p1⊃[s3,t1]¬p1)
axiom W3 [F,t1](¬p2⊃[s1,t1]¬p2)
axiom W3 [F,t1](¬p2⊃[s3,t1]¬p2)
axiom W3 [F,t1](¬p3⊃[s1,t1]¬p3)
axiom W3 [F,t1](¬p3⊃[s2,t1]¬p3)
axiom W4 p1⊃[s2,t1]p1
axiom W4 p1⊃[s3,t1]p1
axiom W4 p2⊃[s1,t1]p2
axiom W4 p2⊃[s3,t1]p2
axiom W4 p3⊃[s1,t1]p3
axiom W4 p3⊃[s2,t1]p3
axiom W4 ¬p1⊃[s2,t1]¬p1
axiom W4 ¬p1⊃[s3,t1]¬p1
axiom W4 ¬p2⊃[s1,t1]¬p2
axiom W4 ¬p2⊃[s3,t1]¬p2
axiom W4 ¬p3⊃[s1,t1]¬p3
axiom W4 ¬p3⊃[s2,t1]¬p3
```

The W3 axioms express the common belief that each of the wise men hold about each other's perceptual abilities. W4 states that these abilities actually hold in the real world.

```
axiom L1 [s1,t1]p1⊃[F,t2][s1,t1]p1
axiom L1 ¬[s1,t1]p1⊃[F,t2]¬[s1,t1]p1

axiom L2 [s2,t2]p2⊃[F,t3][s2,t2]p2
axiom L2 ¬[s2,t2]p2⊃[F,t3]¬[s2,t2]p2
```

L1 and L2 are the learining axioms. L1 states that whatever s1 believes about his spot in situation t1 becomes a common belief in situation t2, *i.e.*, s1 has communicated this fact to the others. L2 does the same for s2 in situation t2.

```
axiom PK1 p2 ∧ p3
axiom C1 <s1:W2,W3,W4,PK1>p1 ≡ [s1,t1]p1

axiom SK1 [s1,t1]p1
axiom SK2 ¬[s1,t1]p1
axiom PK2 p1 ∧ p3

axiom C2 [s1,t1]p1 ⊃  (<s2:W2,W3,W4,L1,SK1,PK2>p2 ≡ [s2,t2]p2)
axiom C2 ¬[s1,t1]p1 ⊃  (<s2:W2,W3,W4,L1,SK2,PK2>p2 ≡ [s2,t2]p2)
```

C1 and C2 are the axioms that describe the circumscription of beliefs for the agents s1 in situation t1 and s2 in t2. If s1 can infer p1 from W2, W3, W4, and the two white spots that he sees, then he knows the color of his own spot; and if he cannot, then he does not know it. In t2, the situation is more complicated, because s2 has learned that s1 either knows the color of his own spot, or does not. The statement of circumscription depends on which of these actually obtains, and the axiom C2 takes each of the possible outcomes into account. Note that this axiomatization is essentially the same as the one in Section 6.3.

The clause form of the axioms is as follows:

```
W1:  p1 ∧ (p2 ∧ p3)
W2:  [F,t1](p1 ∨ (p2 ∨ p3))
W3:  [F,t1](p1 ⊃ ([s2,t1]p1))
W3:  [F,t1](p1 ⊃ ([s3,t1]p1))
W3:  [F,t1](p2 ⊃ ([s1,t1]p2))
W3:  [F,t1](p2 ⊃ ([s3,t1]p2))
W3:  [F,t1](p3 ⊃ ([s1,t1]p3))
W3:  [F,t1](p3 ⊃ ([s2,t1]p3))
W3:  [F,t1](¬p1 ⊃ ([s2,t1]¬p1))
W3:  [F,t1](¬p1 ⊃ ([s3,t1]¬p1))
W3:  [F,t1](¬p2 ⊃ ([s1,t1]¬p2))
W3:  [F,t1](¬p2 ⊃ ([s3,t1]¬p2))
W3:  [F,t1](¬p3 ⊃ ([s1,t1]¬p3))
W3:  [F,t1](¬p3 ⊃ ([s2,t1]¬p3))
W4:  p1 ⊃ ([s2,t1]p1)
W4:  p1 ⊃ ([s3,t1]p1)
W4:  p2 ⊃ ([s1,t1]p2)
W4:  p2 ⊃ ([s3,t1]p2)
W4:  p3 ⊃ ([s1,t1]p3)
W4:  p3 ⊃ ([s2,t1]p3)
W4:  ¬p1 ⊃ ([s2,t1]¬p1)
W4:  ¬p1 ⊃ ([s3,t1]¬p1)
W4:  ¬p2 ⊃ ([s1,t1]¬p2)
W4:  ¬p2 ⊃ ([s3,t1]¬p2)
W4:  ¬p3 ⊃ ([s1,t1]¬p3)
W4:  ¬p3 ⊃ ([s2,t1]¬p3)
L1:  ([s1,t1]p1) ⊃ ([F,t2]([s1,t1]p1))
L1:  ¬[s1,t1]p1 ⊃ ([F,t2]¬[s1,t1]p1)
```

```
L2:  ([s2,t2]p2) ⊃ ([F,t3]([s2,t2]p2))
L2:  ¬[s2,t2]p2 ⊃ ([F,t3]¬[s2,t2]p2)
PK1:  p2 ∧ p3
C1:  (<s1:W2,W3,W4,PK1>p1) ≡ ([s1,t1]p1)
SK1:  [s1,t1]p1
SK2:  ¬[s1,t1]p1
PK2:  p1 ∧ p3
C2:  ([s1,t1]p1) ⊃ ((<s2:W2,W3,W4,L1,SK1,PK2>p2) ≡ ([s2,t2]p2))
C2:  ¬[s1,t1]p1 ⊃ ((<s2:W2,W3,W4,L1,SK2,PK2>p2) ≡ ([s2,t2]p2))
```

These clauses, along with the goal clause

```
Goal:  ¬[s3,t3]p3
```

were submitted to an implementation of the *DPP* procedure on a Symbolics 3600 Lisp machine. The resultant proof took approximately 3 seconds. The trace of the proof follows, showing the 15 recursive calls to the procedure that were generated. No splits were necessary.

```
1.  Entering DPP level ()
  Attaching to circ lit <s2:W2,W3,W4,L1,SK2,PK2>p2 in clause C2
 2.  Entering DPP level (CIRC)
 2.  At level (CIRC) with negative modal atoms:
  SK2:  ¬[s1,t1]p1
   Attaching to modal atom SK2:  ¬[s1,t1]p1
  3.  Entering DPP level ((s1 t1) CIRC)
  3.  Satisfied!  at level ((s1 t1) CIRC)
 2.  At level (CIRC) Satisfied attachment to SK2:  ¬[s1,t1]p1
 2.  Satisfied!  at level (CIRC)
  False circ lit <s2:W2,W3,W4,L1,SK2,PK2>p2 in clause C2
  Attaching to circ lit ¬<s2:W2,W3,W4,L1,SK2,PK2>p2 in clause C2
 4.  Entering DPP level (CIRC)
 4.  At level (CIRC) with negative modal atoms:
  SK2:  ¬[s1,t1]p1
   Attaching to modal atom SK2:  ¬[s1,t1]p1
  5.  Entering DPP level ((s1 t1) CIRC)
  5.  Satisfied!  at level ((s1 t1) CIRC)
 4.  At level (CIRC) Satisfied attachment to SK2:  ¬[s1,t1]p1
 4.  Satisfied!  at level (CIRC)
```

```
  True circ lit ¬<s2:W2,W3,W4,L1,SK2,PK2>p2 in clause C2
  Attaching to circ lit <s2:W2,W3,W4,L1,SK1,PK2>p2 in clause C2
 6.  Entering DPP level (CIRC)
 6.  Satisfied!  at level (CIRC)
  False circ lit <s2:W2,W3,W4,L1,SK1,PK2>p2 in clause C2
  Attaching to circ lit ¬<s2:W2,W3,W4,L1,SK1,PK2>p2 in clause C2
 7.  Entering DPP level (CIRC)
 7.  Satisfied!  at level (CIRC)
  True circ lit ¬<s2:W2,W3,W4,L1,SK1,PK2>p2 in clause C2
  Attaching to circ lit <s1:W2,W3,W4,PK1>p1 in clause C1
 8.  Entering DPP level (CIRC)
 8.  Satisfied!  at level (CIRC)
  False circ lit <s1:W2,W3,W4,PK1>p1 in clause C1
1.  At level () with negative modal atoms:
 C2:  ¬[s2,t2]p2
 C1:  ¬[s1,t1]p1
 GOAL: ¬[s3,t3]p3
  Attaching to modal atom C2:  ¬[s2,t2]p2
 9.  Entering DPP level ((s2 t2))
 9.  At level ((s2 t2)) with negative modal atoms:
  L1:  ¬[s1,t1]p1
   Attaching to modal atom L1:  ¬[s1,t1]p1
  10.  Entering DPP level ((s1 t1) (s2 t2))
  10.  Satisfied!  at level ((s1 t1) (s2 t2))
 9.  At level ((s2 t2)) Satisfied attachment to L1:  ¬[s1,t1]p1
 9.  Satisfied!  at level ((s2 t2))
1.  At level () Satisfied attachment to C2:  ¬[s2,t2]p2
  Attaching to modal atom C1:  ¬[s1,t1]p1
 11.  Entering DPP level ((s1 t1))
 11.  Satisfied!  at level ((s1 t1))
1.  At level () Satisfied attachment to C1:  ¬[s1,t1]p1
  Attaching to modal atom GOAL: ¬[s3,t3]p3
 12.  Entering DPP level ((s3 t3))
 12.  At level ((s3 t3)) with negative modal atoms:
  L1:  ¬[s1,t1]p1
  L2:  ¬[s2,t2]p2
   Attaching to modal atom L1:  ¬[s1,t1]p1
  13.  Entering DPP level ((s1 t1) (s3 t3))
  13.  Satisfied!  at level ((s1 t1) (s3 t3))
 12.  At level ((s3 t3)) Satisfied attachment to L1:  ¬[s1,t1]p1
   Attaching to modal atom L2:  ¬[s2,t2]p2
  14.  Entering DPP level ((s2 t2) (s3 t3))
```

```
     14.  At level ((s2 t2) (s3 t3)) with negative modal atoms:
      L1:  ¬[s1,t1]p1
       Attaching to modal atom L1:  ¬[s1,t1]p1
      15.  Entering DPP level ((s1 t1) (s2 t2) (s3 t3))
      15.  UnSatisfied!  at level ((s1 t1) (s2 t2) (s3 t3))
     14.  At level ((s2 t2) (s3 t3)) Failed attachment to L1:  ¬[s1,t1]p1
     14.  UnSatisfied!  at level ((s2 t2) (s3 t3))
    12.  At level ((s3 t3)) Failed attachment to L2:  ¬[s2,t2]p2
    12.  UnSatisfied!  at level ((s3 t3))
   1.  At level () Failed attachment to GOAL: ¬[s3,t3]p3
   1.  UnSatisfied!  at level ()
```

The numbers refer to a particular call to *DPP*. The basic structure of the proof is as follows. The procedure is first entered (call 1), and the reduction rules *Single* and *Detach* applied to eliminate as many clauses as possible. Next all the circumscriptive literals are evaluated; this produces calls 2–8 to *DPP* in the proof trace. The level of the call is simply the chain of attachments that led to the call. Hence a level of `((s1 t1) Circ)` means that the call was generated first by an attachment to a circumscription literal, then by attachment to the beliefs of agent **s1** in situation **t1**. Looking at calls 2 and 3, we find a typical sequence. In trying to find whether the circumscriptive literal in clause **C2** is valid or not, a recursive call to *DPP* is made. The reduction rules apply, and finally an irreducible set of belief literals is obtained (there is no splitting). Each of the negative belief literals of the set must be checked for satisfiability: there is only one, and *DPP* is called on it (call 3). The negative belief literal **¬[s1,t1]p1** is satisfied because **s1** indeed does not know the color of his own spot in **t1**. Thus the only negative literal remaining in call 2 is satisfied, and so the call returns the information that its input can be satisfied. This means that the circumscriptive literal is always false, and can be eliminated from the clause **C2** in which it occurs.

The proof through call 8 determines that **s1** and **s2** are ignorant of their own spot's color: thus the singleton clauses **¬[s1,t1]p1** and **¬[s2,t2]p2** have been deduced from the circumscriptive axioms **C1** and **C2**, respectively. At this point there is only an irreducible set of belief literals that must be checked for satisfiability, and the proof trace returns to the first call to *DPP*, showing what negative modal atoms exist. Each of these is checked for satisfiability by calling the *DPP* procedure recursively: the first at call 9, the second at call 11, and the third at call 12. This last call is for the goal literal ¬**[s3,t3]p3**, which is found to be inconsistent with the rest of the belief literals. It is interesting to see that the reasoning needed to find the inconsistency involves attaching to **s3**'s view of **s2**'s view of **s1**'s beliefs, just as in the sequent proof in Section 6.3.

References

[1] Appelt, D. E., "Planning Natural-Language Utterances to Satisfy Multiple Goals," *SRI Artificial Intelligence Center Technical Note 259*, SRI International, Menlo Park, California (1982).

[2] Barwise, J. and Perry, J., *Situations and Attitudes*, MIT Press, Cambridge, Massachusetts, 1983.

[3] Boolos, G., *The Unprovability of Consistency*, Cambridge University Press, 1979.

[4] Brachman, R., "Recent Advances in Representational Languages," Invited lecture at the National Conference on Artificial Intelligence, Stanford University, Stanford, California (1980).

[5] Chang, C. L. and Lee, R. C. T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[6] Clark, H., "Responding to Indirect Speech Acts," *Cognitive Psychology* **11**, 4 (1979).

[7] Collins, A. M., Warnock, E., Aiello, N. and Miller, M., "Reasoning from Incomplete Knowledge," in *Representation and Understanding*, Bobrow, D. G., and Collins, A. (eds.), Academic Press, New York (1975).

[8] Creary, L. G., "Propositional attitudes: Fregean representation and simulative reasoning," *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Tokyo (1979), pp. 176–181.

[9] Davis, M., and Putnam, H., "A Computing Procedure for Quantification Theory," *Journal of the Association for Computing Machinery* **7**, 3 (1960), pp. 201-215.

[10] Doyle, J., "Truth Maintenance Systems for Problem Solving," *Artificial Intelligence Laboratory Technical Report 419*, Massachusetts Institute of Technology, Cambridge, Massachusetts (1978).

[11] Fariñas-del-Cerro, L., *Deduction Automatique et Logique Modale,* Thèse d'Etat, Laboratoire Informatique Théorique et Programmation, Université Paris VII, Paris, France, 1981.

[12] Fariñas-del-Cerro, L., "Temporal Reasoning and Termination of Programs," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Universität Karlsruhe, Karlsruhe, West Germany (1983), pp. 926–929.

[13] Fodor, J. A., *The Language of Thought,* Thomas Y. Cromwell Company, New York, New York, 1975.

[14] Goad, C., "A Formal Representation for Situations Involving Knowledge," *unpublished note*, Stanford University, Stanford, California (1976).

[15] Gilmore, P., "The Consistency of Partial Set Theory without Extensionality," in *Axiomatic Set Theory,* T. Jech (ed.), American Math Society (1974), pp. 147–153.

[16] Green, C., "Theorem Proving by Resolution as a Basis for Question-Answering Systems," in *Machine Intelligence 4,* D. Michie and B. Meltzer (eds.), Edinburgh University Press, Edinburgh, Scotland (1969).

[17] Haspel, C. H., *A Study of some Interpretations of Modal and Intuitionistic Logics in the First Order Predicate Calculus,* Doctoral dissertation, Syracuse University, Syracuse, New York, 1972.

[18] Hayes, P. J., "In Defence of Logic," *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Massachusetts Institute of Technology, Cambridge, Massachusetts (1977), pp. 539–565.

[19] Hewitt, C., *Description and Theoretical Anlysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot,* Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1972.

[20] Hintikka, J., "Form and Content in Quantification Theory," *Acta Philsophica Fennica* **8** (1955), pp. 7–55.

[21] Hintikka, J., *Knowledge and Belief,* Cornell University Press, Ithaca, New York, 1962.

[22] Hintikka, J., "Semantics for Propositional Attitudes," in *Reference and Modality*, L. Linsky (ed.), Oxford University Press, London (1971), pp. 145–167.

[23] Hughes, G. E. and Cresswell, M. J., *Introduction to Modal Logic,* Methuen and Company Ltd., London, England, 1968.

[24] Israel, D. J., "What's Wrong with Non-Monotonic Logic?," *Proceedings of the First National Conference on Artificial Intelligence*, Stanford University, Stanford, California (1980).

[25] Johnson-Laird, P. N. and Steedman, M., "The Psychology of Syllogisms," *Cognitive Psychology* **10**, 1 (1978), pp. 64–100.

[26] Johnson-Laird, P. N., "Mental Models in Cognitive Science," *Cognitive Science* **4** (1980), pp. 71-115.

[27] Kleene, S. C., *Introduction to Metamathematics*, D. Van Nostrand Company, Princeton, New Jersey, 1952.

[28] Kleene, S. C., *Mathematical Logic*, John Wiley and Sons, New York, 1967.

[29] König, D., "Sur les correspondences multivoques des ensembles," *Fundamenta Mathematicæ* **8** (1926), pp. 114-34.

[30] Konolige, K., "A First Order Formalization of Knowledge and Action for a Multiagent Planning System," in *Machine Intelligence 10*, J. E. Hayes, D. Michie, and Y-H Pao (eds.), Ellis Horwood Limited, Chichester, England (1982).

[31] Konolige, K., "Circumscriptive Ignorance," *Proceedings of the Second National Conference on Artificial Intelligence*, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1982).

[32] Konolige, K., "Modal Logics for Belief," *unpublished note* (1982).

[33] Konolige, K., "A Deductive Model of Belief," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Universität Karlsruhe, Karlsruhe, West Germany (1983).

[34] Kowalski, R., *Logic for Problem Solving*, North-Holland, New York, 1979.

[35] Kripke, S. A., "A Completeness Theorem in Modal Logic," *Journal of Symbolic Logic* **24** (1959), pp. 1–15.

[36] Kripke, S. A., "Semantical Considerations on Modal Logic," *Acta Philosophica Fennica* **16** (1963), pp. 83–94.

[37] Kripke, S. A., "Naming and Necessity," in *Semantics of Natural Language*, D. Davidson and G. Harmon (eds.), D. Reidel Publishing Company, Dordrecht, Holland (1972), pp. 253–355.

[38] Levesque, H. J., "A Formal Treatment of Incomplete Knowledge Bases," *FLAIR Technical Report No. 614*, Fairchild, Palo Alto, California (1982).

[39] Levesque, H. J., "A Logic of Knowledge and Active Belief," *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, Texas (1984).

[40] Lycan, W. G., "Toward a Homuncular Theory of Believing," *Cognition and Brain Theory* **4**, 2 (1981), pp. 139–59.

[41] Maida, A. S., "Knowing Intensional Individuals, and Reasoning about Knowing Intensional Individuals," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Universität Karlsruhe, Karlsruhe, West Germany (1983), pp. 382–384.

[42] McCarthy, J., "Towards a Mathematical Science of Computation," *Information Processing, Proceedings of the IFIP Congress* **62**, North-Holland Publishing Company, Amsterdam (1962), pp. 21–28.

[43] McCarthy, J., Sato, M., Hayashi, T., and Igarashi, S., "On the Model Theory of Knowledge," *Stanford Artificial Intelligence Laboratory Memo AIM-312*, Stanford University, Stanford (1978).

[44] McCarthy, J., "Formalization of two puzzles involving knowledge," *unpublished note*, Stanford University, Stanford, California (1978).

[45] McCarthy, J., and Hayes, P. J., "Some Philosophical Problems form the Standpoint of Artificial Intelligence," in *Machine Intelligence* **4**, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, Scotland (1969), pp. 463–502.

[46] McCarthy, J., "First Order Theories of Individual Concepts and Propositions," in *Machine Intelligence* **9**, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, Scotland (1979), pp. 120–147.

[47] McCarthy, J., "Circumscription—A Form of Non-Monotonic Reasoning," *Artificial Intelligence* **13**, 1–2 (1980).

[48] McDermott, D. and Doyle, J., "Non-Monotonic Logic I," *Artificial Intelligence* **13**, 1–2 (1980).

[49] Montague, R., "Syntactical Treatments of Modality, with Corollaries on Reflexion Principles and Finite Axiomatizability," *Acta Philosophica Fennica* **16** (1963), pp. 153–67.

[50] Moore, R. C., "Reasoning from Incomplete Knowledge in a Procedural Deduction System," *MIT Artificial Intelligence Laboratory, AI-TR-347* (1975).

[51] Moore, R. C., "Reasoning About Knowledge and Action," *Artificial Intelligence Center Technical Note 191*, SRI International, Menlo Park, California (1980).

[52] Moore, R. C, "Semantical Considerations on Nonmonotonic Logic," *SRI Artificial Intelligence Center Technical Note 284*, SRI International, Menlo Park, California (1983).

[53] Moore, R. C. and Hendrix, G. G., "Computational Models of Belief and the Semantics of Belief Sentences," *SRI Artificial Intelligence Center Technical Note 187*, SRI International, Menlo Park, California (1979).

[54] Nilsson, N., *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, California, 1980.

[55] Perlis, D., "Language, Comptation, and Reality," *Department of Computer Science TR95*, University of Rochester, Rochester, New York (1981).

[56] Perlis, D., "True Beliefs I," *unpublished ms.* (1984).

[57] Perry, J., "The Problem of the Essential Indexical," *NOÛS* **13** (1979), pp. 3–21.

[58] Reiter, R., "A Logic for Default Reasoning," *Artificial Intelligence* **13**, 1–2 (1980).

[59] Robinson, J. A., *Logic: Form and Function*, Elsevier North Holland, New York, 1979.

[60] Robinson, J. A., "A Machine-Oriented Logic Based on the Resolution Principle," *Journal of the Association for Computing Machinery* **12** (1965), pp. 23–41.

[61] Sato, M., *A Study of Kripke-type Models for Some Modal Logics by Gentzen's Sequential Method*, Doctoral dissertation, Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan, 1976.

[62] Schubert, L. K., "Extending the Expressive Power of Semantic Nets," *Artificial Intelligence* **7**, 2 (1976), pp. 163–198.

[63] Smullyan, R. M., *First-Order Logic*, Springer-Verlag, New York, 1968.

[64] Stallman, R. M. and Sussman, G. J., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *MIT Artificial Intelligence Laboratory, AI-TR-297* (1976).

[65] Warren, D. H. D., "WARPLAN: A System for Generating Plans," *Dept. of Computational Logic Memo 76*, University of Edinburgh School of Artificial Intelligence, Edinburgh, Scotland (1974).

[66] Weyhrauch, R., "Prolegomena to a Theory of Mechanized Formal Reasoning," *Artificial Intelligence* **13** (1980).

[67] Winograd, T., "Extended Inference Modes in Reasoning by Computer Systems," *Artificial Intelligence* **13**, 1–2 (1980).

[68] Woods, W., "What's in a Link?," in *Representation and Understanding*, Bobrow, D. G., and Collins, A. (eds.), Academic Press, New York (1975).