SIMBA: Belief Ascription by Way of Simulative Reasoning

by

Hans Chalupsky

A dissertation submitted[1] to the
Faculty of the Department of Computer Science
and the Graduate School of the
State University of New York at Buffalo
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Dissertation Committee:

Dr. Stuart C. Shapiro, Chairman
Dr. Bharadwaj Jayaraman
Dr. William J. Rapaport

January, 1996

[1]This is a single-spaced reprint of the originally submitted version.

Abstract

A key cognitive faculty that enables humans to communicate with each other is their ability to incrementally construct and use models describing the mental states of others, in particular, models of their beliefs. Not only do humans have beliefs about the beliefs of others, they can also reason with these beliefs even if they do not hold them themselves. If we want to build an artificial or computational cognitive agent that is similarly capable, we need a formalism that is fully adequate to *represent* the beliefs of other agents, and that also specifies how to *reason* with them.

Standard formalizations of knowledge or belief, in particular the various epistemic and doxastic logics, seem to be not very well suited to serve as the formal device upon which to build an actual computational agent. They neglect either representation problems, or the reasoning aspect, or the defeasibility that is inherent in reasoning about somebody else's beliefs, or they use idealizations which are problematic when confronted with realistic agents.

Our main result is the development of SIMBA, an implemented belief-reasoning engine that uses *simulative reasoning* to reason with and about the beliefs of other agents. SIMBA is built upon **SL**, a fully intensional, subjective, nonmonotonic logic of belief which is representationally and inferentially adequate to serve as one of the main building blocks of an artificial cognitive agent. **SL** can handle agents that do not believe the consequential closure of their base beliefs; it is adequate to model introspection; it facilitates belief maintenance and revision; and it has a more intuitive semantics than standard formalizations of belief.

Für meine Eltern

Acknowledgments

Even though a dissertation is to be written by one person alone, it really takes the help and support of many people to successfully finish it. First of all, I want to thank my advisor Stu Shapiro for taking me on as a student, for providing a very interesting research environment within the SNePS research group, for his long-term goal and vision of constructing an artificial cognitive agent called Cassie, and for his incredible persistence in pursuing it. I also want to thank him for all his constructive criticism and advice, his patience with my graduate student work ethic, and his way out whenever I got stuck. If there are any good ideas in this dissertation, chances are they are his. He once joked "all research is done by faculty — just under very trying circumstances", I think I now know what he meant.

I want to thank Bill Rapaport for his many valuable suggestions and comments, for his inexhaustible list of relevant literature, for his careful reading of my sloppy drafts, for his incredibly short turn-around time, and generally for providing the watchful and critical eye of the philosopher. Thanks also to Bharat Jayaraman for his valuable comments, and for providing a fresh and different perspective. Many thanks to my outside reader John Barnden for his careful reading and valuable suggestions, clarifications, and criticisms. Finally, thanks to all the members of the SNePS research group for their helpful questions and comments, and to my whole committee for being so extremely tolerant of my bad time management.

Thanks to the department secretaries for all their help over the years, but in particular to Ellie who probably worried more about my graduation than I ever did.

Thanks to my very good friends and ex-colleagues Deepak Kumar and Richard Wyatt for showing me that it can be done, and for telling me over and over again to "just do it". Thanks also to the Molson brewery for providing endless food for thought, and to Bobby and Debbie Brown, Phil, and Jack, for selling it to me.

Many many thanks to my girlfriend Molly for her unconditional, loving support, and for putting up with my work hours that were more befitting of a vampire.

Most of all, thanks to my parents for their unconditional love and never-ending faith in my abilities. Dad, you couldn't wait long enough for me to finish, but wherever you are I hope you can still celebrate "mit an guaten Viertel Tattendorfer".

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A key cognitive faculty that enables humans to communicate with each other in a fruitful and meaningful way is the ability to incrementally construct and use models describing the mental states of others, for example, what they believe, intend, plan, hope, etc. A crucial component of such *agent models* is the part which describes beliefs, and it is that aspect with which we will primarily be concerned. This is not to say that the modeling of intentions, plans, desires, hopes, etc., is not as important, it is just that these concepts are even less understood than belief, and we simply hope that our model of belief developed below will be able to peacefully coexist with future treatments of these other important cognitive phenomena.

It should be stressed right away that by *agent model* we typically mean a part of a cognitive agent that models aspects of some other agent that the former agent knows about; it does not refer to the artificial cognitive agent as a whole, which in itself constitutes a model of an agent.

An agent model will realistically contain only a finite number of sentences expressed in some language of thought. Hence, assuming sufficiently powerful inference rules, some of its consequences will remain implicit. To make them explicit, the agent holding the model has to use some form of reasoning. One possible reasoning strategy applicable to this problem can be paraphrased as "what would I believe if I were in the other agent's position", or more precisely, "what would I believe if I were the other agent believing exactly what I believe that agent believes". Such a strategy can be viewed as a *simulation* in which an agent uses its model of some other agent in conjunction with its own reasoning abilities to simulate the other agent's reasoning.

If the task at hand is to build an artificial (or computational) cognitive agent capable of communicating with other agents, be they artificial or human, then one must understand how to construct such agent models, and how to connect and integrate them with a reasoning mechanism that can make implicit information explicit. We claim that a certain kind of *propositional belief model* in conjunction with a *simulative reasoning* strategy of the kind described above can be used successfully to achieve that goal.

Any reasonable attempt to construct an artificial cognitive agent — in the following, we will refer to it as Cassie — will contain at least the following two modules: a knowledge base and a reasoning mechanism. The knowledge base is a database that explicitly describes by way of some formalism things[1] that are assumed to be known or believed[2] by Cassie. Such a knowledge base could be, for example, a set of sentences of first-order logic or a semantic network of some sort.

[1]Such "things" can be anything that may be imagined by a mind, not just things that exist in the real world. See, for example, [Shapiro and Rapaport, 1991] for arguments in favor of this view.

[2]We will use the terms *knowledge* and *belief* rather interchangeably, even though technically, we will always mean

The reasoning mechanism is necessary to make information that is only implicit in the knowledge base explicit. For example, Cassie might know that dogs are animals, and then learn during a conversation that Fido is a dog. From that, she should be able to infer that Fido is an animal. Any state-of-the-art knowledge representation system can handle this standard sort of inference. Now consider the case that Cassie knows some other agent Lucy, and Lucy tells Cassie: "Fido is a dog". This utterance might prompt Cassie to add the following to her knowledge base: Lucy believes that (some) Fido is a dog. If we then ask Cassie whether Lucy believes that (some) Fido is an animal, she should in most circumstances answer yes, even though she did not have an explicit belief about that. The reason we said "(some) Fido" here is that Cassie might not know any dog named Fido, or she might believe that the one Lucy seems to refer to is different from all the Fidos she knows.

The two forms of inference described above look very similar. However, the reasoning in the second case has to be quite different than in the first case. Assuming that Cassie employs simulative reasoning she had to go through the following steps: First, she had to hypothetically assume Lucy's belief that (some) Fido is a dog, even though she herself might not know a dog named Fido or might believe that Fido is actually a cat that just looks very much like a dog. Then Cassie had to attribute the belief that dogs are animals to Lucy on the strength of the assumption that this is common knowledge known by just about anyone. Next, she had to simulate Lucy's reasoning by attributing her own reasoning strategies to Lucy, thus leading to the conclusion that from Lucy's beliefs it follows by simulation that (some) Fido is an animal. Finally, she could ascribe the conclusion of the simulation to Lucy as another belief explicitly stored in her model of Lucy.

A computational model for reasoning of the kind described above has to account for at least the following problems:

- Models of other agents have to be used *accurately*, for example, none of Cassie's private beliefs should be accidentally attributed to other agents.

- Models of other agents have to be generated *automatically*. Just as in the simple common knowledge attribution above, arbitrary reasoning might be necessary to make various parts of an agent model explicit before they can be used in the simulation of that agent. This process will be called *meta-reasoning*, since it stands outside the simulated agent's reasoning and is explicitly about it, rather than being within the simulation itself.

- The simulative reasoning mechanism should be closely integrated with the agent's own reasoning.

- Simulation has to work at arbitrary levels of nesting, for example, if Cassie wants to simulate Lucy's simulation of John, or Lucy's simulation of herself, Cassie.

- The model has to account for the *defeasibility* that is inherent in this kind of reasoning, because in general we can never be sure whether the result of a simulation is actually believed by the simulated agent.

A considerable amount of work in artificial intelligence, philosophy, and other cognitive sciences has been concerned with the formalization and modeling of the phenomenon of belief. Such formalisms are usually called *belief systems*, and they are great in number; see, for example, [Hintikka, 1962; Creary, 1979; Levesque, 1984; Moore, 1980; Wilks and Bien, 1983; Konolige, 1986; Rapaport,

belief and not at all be concerned with truth, which is generally viewed as a necessary ingredient of knowledge. For us a false belief is as good as any other.

1986; Rapaport, 1992; Ballim and Wilks, 1991] for a small selection. On the formal end of the spectrum are the various epistemic and doxastic logics, many of them modal in nature and inspired by Hintikka's famous treatment of knowledge and belief [Hintikka, 1962]. On the more psychologically oriented end are systems such as Viewgen [Ballim and Wilks, 1991], or ViewFinder [Ballim, 1992].

We feel, however, that none of these formalisms is fully adequate to serve as the foundation upon which to build an actual computational agent. They neglect either representation problems, or the reasoning aspect, or the defeasibility that is inherent in reasoning about somebody else's beliefs, or they use idealizations which are problematic when applied to realistic agents. Of course, what constitutes a *proper* or *good* model or formalism can hardly be decided by theoretical considerations alone, it has to be determined by experimentation in actual applications. Thus one of our main concerns is to provide an implementation of our model on a computer, so that its utility can be studied in practice.

Chapter 2

Simba: A Subjective, Simulative, Multi-Contextual Belief Reasoner

Having described the overall problem in very general terms, we will now examine it in more detail and give an outline of our solution called Simba. Simba is an acronym for **sim**ulative **b**elief **a**scription. It is an implemented belief-reasoning engine intended to serve as one of the main building blocks of the mind of our computational agent Cassie. Simba enables Cassie to reason with her own beliefs about the — not necessarily extensional — world around her, but the main emphasis of Simba is to give Cassie the ability to reason with and about other agents' beliefs, or, more precisely, to reason with beliefs she believes are held by other agents. The sections below provide an overview of the basic agent model underlying Simba and motivate its various features.

2.1 The Belief Model

What does it mean for a computational agent such as Cassie to believe something? How does she do that? What *is* belief anyway? These are questions that need to be addressed by a belief model such as the one underlying Simba.

One way of characterizing the phenomenon of human agents having beliefs is to say that these beliefs are some semi-permanent aspect of their neuronal or mental state that makes them act a certain way, for example, answer "yes" when they are asked whether it was raining yesterday. According to this theory, if an agent has a belief such as "yesterday it was raining", then there must be some actual brain tissue of the agent in some special state, such that without that tissue (or by not being in a special state) the agent would not hold that belief.

A simple way to apply this characterization to Cassie is to think of her mind as a container that is filled with some sort of "objects", some of which we will call her beliefs. Such a set of objects believed by an agent is frequently called a *belief space*. The question is: What are these "belief objects", and how can they be described? Nobody knows what belief really *is*, but an appealing characterization that is often used is that belief is a *propositional attitude*, that is a relation between an agent and a proposition. Taking this characterization seriously, we will define a language whose expressions denote propositions, and we will construct Cassie's mind as a set of such expressions.

Depending on a particular implementation or storage scheme, not all expressions in Cassie's mind will denote propositions actually believed by her. Some might just be residue from pondering certain questions, others might be part of other propositions. To be able to single out Cassie's actual beliefs, we will take every expression in her mind to be associated with a flag, which, if it is on, will indicate that the proposition denoted by that expression is actually believed by Cassie.

Following logic tradition, we will use a formal logical language to describe Cassie's beliefs. The language is called $L_{\mathbf{SL}}$, since it forms part of a logic called **SL**. **SL** stands for SIMBA Logic. It constitutes the logical foundation of SIMBA. The other two major components of **SL** are a semantic account and a deductive system. All three components of **SL** are described in detail in Chapter 4.

For this exposition it suffices to say that $L_{\mathbf{SL}}$ looks very much like the language of predicate calculus, but that it has a very different semantics. It is primarily a language of proposition-valued function terms such as, for example, $\mathsf{Loves}(\mathsf{John}, \mathsf{Lucy})$ whose intended denotation is the proposition expressed by the sentence *John loves Lucy.* This is in stark contrast with standard model-theoretic semantics where the denotation of a predicate sentence such as $\mathsf{Loves}(\mathsf{John}, \mathsf{Lucy})$ is given as a truth value.

An $L_{\mathbf{SL}}$-sentence is formed by prefixing a proposition term with an exclamation mark as in $!\mathsf{Loves}(\mathsf{John}, \mathsf{Lucy})$. The exclamation mark serves as the flag mentioned above, which indicates belief of a certain proposition. The semantics of a sentence is that the agent whose mind contains it (usually taken to be Cassie) believes the proposition denoted by the proposition term.

Figure 2.1 shows what a particular snapshot of Cassie's mind could look like. Note that in $L_{\mathbf{SL}}$ logical connectives such as $\wedge, \vee, \Rightarrow, \ldots$ are just special cases of proposition-valued functions which are written in infix notation only to enhance readability. The quantifiers $\forall$ and $\exists$ are not themselves

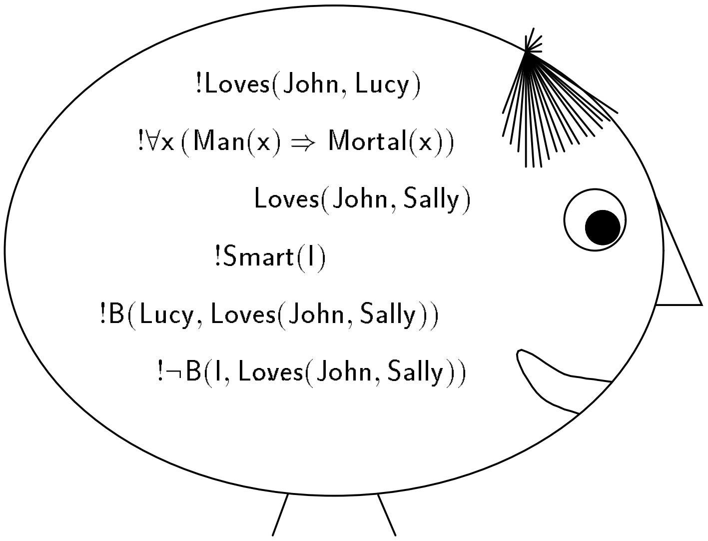


Figure 2.1: Cassie's mind as a container of belief expressions: B is the name of the belief function and I is Cassie's self concept. The intended interpretations of the other symbols should be evident. The '!' is used to flag the propositions believed by Cassie.

functions, but they are also defined as term forming operators which can be used as arguments to functions (cf. Chapter 4). Cassie's beliefs about the beliefs of other agents are expressed by sentences of the kind $!\mathsf{B}(\mathsf{Lucy}, \mathsf{Loves}(\mathsf{John}, \mathsf{Sally}))$. The proposition term of such a sentence is simply a nested application of proposition-valued functions but not a higher-order relation. The nesting can go to arbitrary depth to account for propositions such as *John believes that Sally believes that I believe that....* Note that the top-level expressions are Cassie's immediate beliefs; hence, they need not be framed in an extra application of the belief function B. Only her beliefs about other agent's beliefs as well as her introspective beliefs have to make use of B.

It should be pointed out right away that even though Cassie's beliefs might be viewed as a database of belief sentences, our model is not the database approach to belief representation as described and criticized by Moore [1977]; hence, it does not suffer from the problems of that approach. To form beliefs about the beliefs of other agents Cassie has the full logical arsenal at her disposal, including negation and disjunction. Via introspection, she can even have beliefs about her own beliefs, for example, $!\neg\mathsf{B}(\mathsf{I}, \mathsf{Loves}(\mathsf{John}, \mathsf{Sally}))$. Table 2.1 gives an overview of the representational power of the belief representation language with a set of prototypical examples.

$!\mathsf{Loves}(\mathsf{John}, \mathsf{Lucy})$	...plain belief
$!\forall \mathsf{x}\,(\mathsf{Man}(\mathsf{x}) \Rightarrow \mathsf{Mortal}(\mathsf{x}))$	...quantified belief
$!\mathsf{B}(\mathsf{Lucy}, \mathsf{B}(\mathsf{Sally}, \mathsf{Loves}(\mathsf{John}, \mathsf{Sally})))$	...nested belief
$!\mathsf{B}(\mathsf{I}, \mathsf{Loves}(\mathsf{John}, \mathsf{Lucy}))$	...positive introspection
$!\neg\mathsf{B}(\mathsf{I}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}))$	...negative introspection
$!\mathsf{B}(\mathsf{John}, \mathsf{Nice}(\mathsf{Lucy})) \vee \mathsf{B}(\mathsf{John}, \neg\mathsf{Nice}(\mathsf{Lucy}))$	...disjunctive belief
$!\mathsf{Smart}(\mathsf{I})$	...false belief
$!\forall \mathsf{a}\,(\mathsf{B}(\mathsf{a}, \mathsf{Exists}(\mathsf{Santa})) \Rightarrow \mathsf{Child}(\mathsf{a}))$	...agent quantification
$\mathsf{P} \wedge \neg\mathsf{P}$	...conception without belief

Table 2.1: The representational power of the belief representation language

We are of course well aware that in trying to build an artificial cognitive agent we will have to be concerned with many other important aspects of agenthood such as plans, actions, intentions, sensory apparatus, etc.; however, SIMBA is only concerned with the part that deals with beliefs. For approaches to how inference can be tied in with action, or how new symbols can be introduced by grounding them in perception see, for example, [Kumar, 1994; Lammens, 1994].

It should be evident from the above that we take belief spaces to be sets of propositions rather than sets of sentences. For arguments in favor of this view and against viewing belief as a relation between an agent and a sentence see, for example, [Church, 1950; Shapiro, 1993]. By now, the concerned reader might be worried about the fundamental building block of our theory, propositions, since their nature has been the subject of much philosophical debate. We will not add to this debate. For us, propositions are abstract, intensional entities that are in the domain of discourse, or, as [Creary, 1979, p. 176] puts it, they are "abstractions of things psychological."

2.2 The SNePS Belief Model

The belief model described above is a direct descendant of the SNePS belief model. SNePS, the Semantic Network Processing System developed by Shapiro et al. [Shapiro, 1979; Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992], is a system that implements a fully intensional theory of knowledge representation and reasoning. The main driving force behind its development is the goal to construct an artificial cognitive agent such as Cassie[1] capable of communicating in natural language.

The main difference between our Cassie and the SNePS version of Cassie is the language used to describe her beliefs. While the language of **SL** is linear and very similar to the language of predicate

[1]The name of our prototypical agent is borrowed from the SNePS project. It is an acronym for **C**ognitive **A**gent of the **S**NePS **S**ystem - an **I**ntelligent **E**ntity.

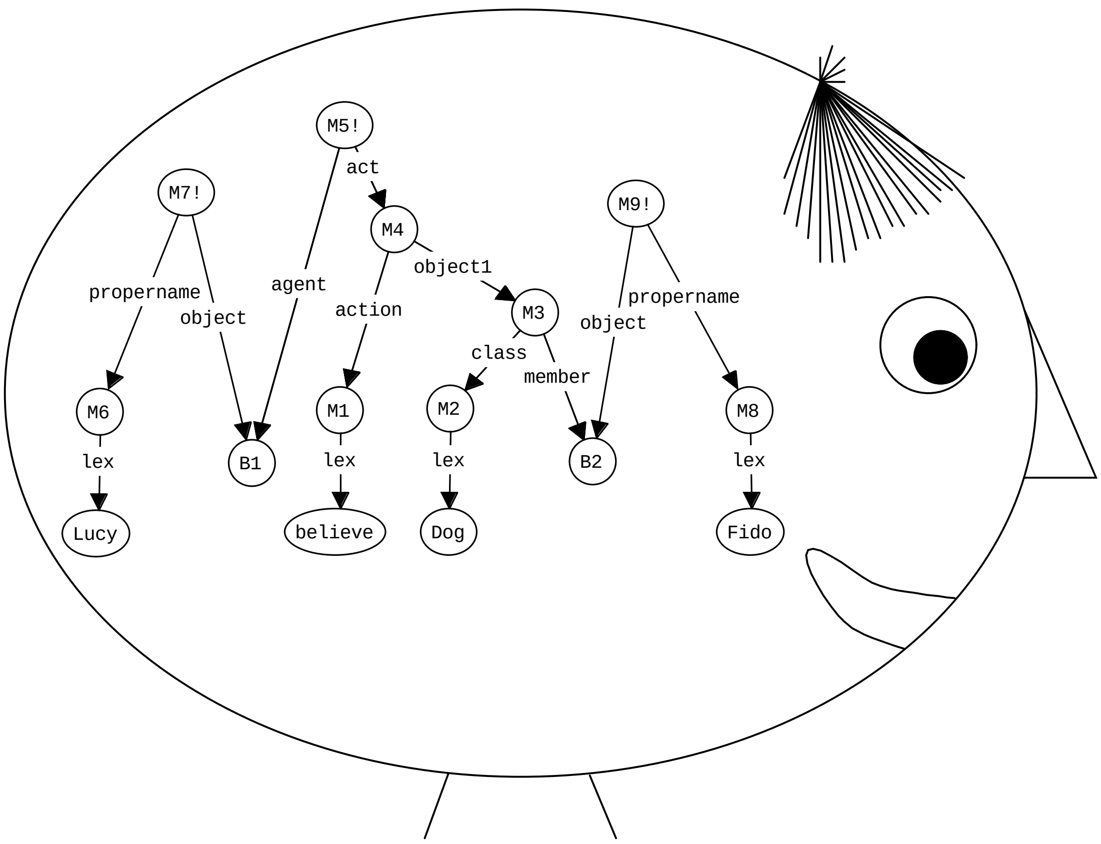


Figure 2.2: SNePS/Cassie's *de re* representation of "Lucy believes that Fido is a dog."

calculus, SNePS is a network language; hence, the cognitive makeup of the SNePS version of Cassie is a semantic network instead of a set of **SL** expressions.

To establish the connection to **SL**, we will briefly describe how Cassie's top-level beliefs as well as her beliefs about other agents are represented in SNePS (for detailed information about various other well-formed SNePS networks and their semantic interpretations, the reader is referred to [Shapiro and Rapaport, 1987]). Suppose Cassie processes the sentence "Lucy believes that Fido is a dog." Figure 2.2 shows how the *de re* interpretation of this sentence is represented as a SNePS network.

This network is a minor variation of one developed by Rapaport [1986]. The only difference is a slightly modified version of the belief case frame used for the central node `M5` which represents Cassie's belief about Lucy's belief.

We will not be concerned with the various problems associated with *de re* and *de dicto* belief reports. For a detailed treatment of these issues, see [Rapaport, 1986; Rapaport *et al.*, 1986; Wiebe and Rapaport, 1986]. Here it suffices to say that the canonical *de re* reading of the above sentence is "Lucy believes *of* Fido that he is a dog" (cf. [Rapaport, 1986, p. 392]), which can be further expanded into "some individual, which Cassie believes is named Lucy, believes of some other individual, which Cassie believes is named Fido, that it belongs to the class of dogs". The crucial part here is that the individual Lucy's belief is about is interpreted *de re*; hence, it is identified with an individual known by Cassie, namely the one she believes is named Fido.

In SNePS, propositions are represented by nodes. For example, the node `M9` represents the proposition that the intensional individual denoted by `B2` is named "Fido". Since this proposition is actually believed by Cassie, `M9` is written with an exclamation mark as a suffix (for aesthetic reasons, **SL** uses the exclamation mark as a prefix operator). Nodes such as `M9` that represent believed propositions are often also called *asserted nodes*. Nodes whose label starts with an `M` are *molecular nodes*, since they have more structure below them that determines their semantics. Molecular nodes roughly correspond to nested function applications in **SL**. Nodes whose names start with a `B` are called *base nodes*. They represent individuals that are further qualified by the propositions asserted about them. Base nodes correspond to individual constants in **SL**. The nodes at the tails of `lex` arcs are not proposition nodes, but what are called *structured individuals*. They represent intensional individuals which are expressed by special *sensory nodes* at the heads of `lex` arcs such as `Fido`. These sensory nodes are links into Cassie's lexicon used by the natural language parser and generator to analyze and generate utterances. Structured individuals correspond to applications of non-propositional functions in **SL**, however, there is no direct analog to sensory nodes, hence, we will also use standard individual constants to represent them.

Let us now briefly explain how this network represents the *de re* reading of the example sentence given above (we will use a standard double-brackets notation to refer to the denotation of a node; for example, $[\![\texttt{B1}]\!]$ is the intensional individual denoted by `B1`): `M7` represents Cassie's belief that some individual $[\![\texttt{B1}]\!]$ is named "Lucy". The central node `M5` represents Cassie's belief that the individual $[\![\texttt{B1}]\!]$ (who she believes is named "Lucy") is engaged in the act of believing the proposition that some individual $[\![\texttt{B2}]\!]$ is a member of the class of dogs. That proposition is represented by the node `M3`, which in this example is not asserted; hence, Cassie does not have any beliefs about whether Fido is a dog. Finally, `M9` represents Cassie's belief that the individual $[\![\texttt{B2}]\!]$ is named "Fido".

Now let us assume that the range of the proposition-valued **SL**-function Name covers the same propositions as are expressed by `object/propername` nodes such as `M9`, that the Member function covers the same propositions as are expressed by `member/class` nodes such as `M3`, and that the non-propositional Lex function has the same semantics as nodes at the tail of `lex` arcs such as `M6`. Then the complete network could be translated into **SL** with the following set of sentences:

$$
\begin{aligned}
&!\mathsf{Name}(\mathsf{B}_1, \mathsf{Lex}(\mathsf{Lucy}))\\
&!\mathsf{B}(\mathsf{B}_1, \mathsf{Member}(\mathsf{B}_2, \mathsf{Lex}(\mathsf{Dog})))\\
&!\mathsf{Name}(\mathsf{B}_2, \mathsf{Lex}(\mathsf{Fido}))
\end{aligned}
$$

The last sentence expresses that Cassie believes the individual Lucy's belief is about (B_2) to be named Fido, which makes this a representation of the *de re* reading as explained above. Wherever possible, however, we will be be sloppy and simply write $\mathsf{B}(\mathsf{Lucy}, \mathsf{Dog}(\mathsf{Fido}))$ instead of a more precise representation that uses Name and Lex functions. The subtleties associated with the representation of *de re* and *de dicto* belief reports can of course not be expressed with such a simplified scheme.

Figure 2.3 shows how Cassie can represent a complete agent model of Lucy in SNePS. Such a model is represented as a set of belief nodes of the kind shown in Figure 2.2. Notice that Cassie might herself believe some of the beliefs she attributes to Lucy. This is illustrated by the asserted proposition $\mathrm{P}_{L1}!$ and the not asserted proposition P_{Lk}. The triangles below these proposition nodes symbolize the networks necessary to represent the propositions. Translating this network into **SL** we would get the following set of sentences:

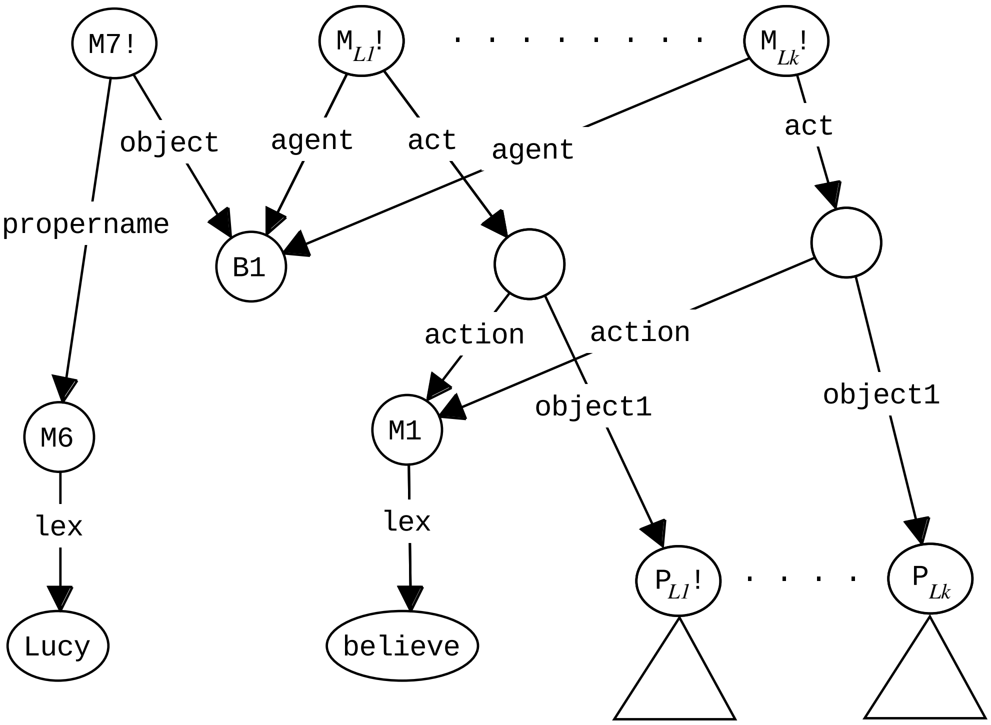


Figure 2.3: SNePS/Cassie's representation of Lucy's beliefs

$!\mathsf{Name}(\mathsf{B}_1, \mathsf{Lex}(\mathsf{Lucy}))$
$!\mathsf{B}(\mathsf{B}_1, \mathsf{P}_{L_1})$
$\ldots$
$!\mathsf{B}(\mathsf{B}_1, \mathsf{P}_{Lk})$
$!\mathsf{P}_{L_1}$
$\ldots$

In summary, the language of **SL** can be seen as SNePS without the network indexing scheme. However, some aspects of SNePS such as path-based inference and order independence of arguments are lost in **SL**. For example, in **SL** $\mathsf{P} \wedge \mathsf{Q}$ denotes a different proposition than $\mathsf{Q} \wedge \mathsf{P}$. This is a consequence of what we call the *uniqueness principle* which is defined and explained in Chapter 4. In SNePS both conjunctions would be represented by the same network, and, hence, have the same denotation which is more desirable. However, since in **SL** we are primarily concerned with the development of a particular reasoning scheme, we will sacrifice some of the representational power of SNePS, in order to be able to use a simpler, predicate-logic-style language that is concise, easy to present, and will allow us to concentrate on the reasoning aspects of the logic **SL**.

2.3 Subjectivity

A unifying characteristic of standard logical treatments of knowledge and belief is that they are, in some aspect or other, quite complicated. Syntactic approaches, for example, [Kaplan, 1968; Haas, 1990], usually employ a quotation device which leads to a notationally complex hierarchy of object and metalanguages. Sentential or modal approaches, for example, [Hintikka, 1962; Levesque, 1982; Halpern and Moses, 1992], commonly use a complex and somewhat unintuitive semantic notion, sets of possible worlds, to interpret belief sentences. Referential opacity of belief contexts, quantifying in, possible falsehood of beliefs, or simply technical difficulties with the formalization make it necessary to complicate things with restricted equality reasoning, standard names, rigid designators, naming maps, etc.

It seems that one of the main sources of these various complexities is that an inherently subjective, intensional phenomenon such as belief gets analyzed in an objective, extensional way. Notions such as *truth* and *possible worlds* are extensional notions, which are then used to objectively analyze the mental states of believers from an outside observer's point of view. Rectifying assertions of truths about the world with assertions about the mental states of believers that are in some way about this world seems to be the main stumbling block. As an alternative, we propose a subjective, intensional model.

SL will be subjective in the sense that it will describe Cassie's subjective view of the world. Expressions of **SL** are not seen as being assertions *about* an agent such as Cassie from an outside observer's point of view; rather, they actually *make up* or *create* the agent. These expressions are the mental substrate of an agent and its only means to represent and reason with beliefs. Thus, the same language that is used to represent Cassie's own believed propositions is used to represent propositions within nested belief contexts. Similarly, individual constants used in belief contexts come from the same mental individual vocabulary that Cassie's own individuals are from. Thus, no technical devices such as standard names or translation maps from one agent's language into another as used by some other belief logics (e.g., [Konolige, 1986]) are necessary. The language of **SL** is Cassie's language of thought, and it is the only language she can think in.

Another aspect of this subjectivity is that Cassie imputes some of her own representation schemes to other agents by the way she represents nested beliefs. For example, suppose Cassie's mind contains the sentence $!\mathsf{B}(\mathsf{Lucy}, \mathsf{B}(\mathsf{Mary}, \mathsf{Loves}(\mathsf{John}, \mathsf{Sally})))$. Here Cassie attributes to Lucy a view of belief identical to her own, namely, that belief is a relation between an agent and a proposition; thus, it might be argued that Cassie imputes a representation of that relation to Lucy along with some representation scheme for propositions. However, since we view this belief model as a good (if not the correct) one, we do not consider this imputation as being particularly harmful.

2.4 The Basic Reasoning Model

The description of the belief model given above showed how to create beliefs in Cassie's head by adding $L_{\mathbf{SL}}$-sentences to it. Each of these sentences can be viewed as one of Cassie's *explicit* beliefs, since there is some actual mental structure associated with every one of them. But what about beliefs that are only *implicit*, that is, beliefs not materialized by any actual sentences in Cassie's brain, but which in some way follow from Cassie's explicit beliefs? For example, if Cassie believes that Fido is a dog, and that dogs are animals, then she should answer "yes" if somebody asks her whether Fido is an animal. This process of making information explicit that before was only available implicitly is usually called *inference* or *reasoning*.

In SIMBA we model reasoning as logical inference. Thus, while the syntax and semantics of **SL** provide the formal basis of Cassie's belief representation, her reasoning gets formalized as logical inference according to a deductive system $D_{\mathbf{SL}}$. An implementation of a proof procedure for $D_{\mathbf{SL}}$ serves as her actual reasoning engine. $D_{\mathbf{SL}}$ is a natural deduction system which consists of a part very similar to natural deduction systems for standard first-order predicate calculus, and a part that deals with belief reasoning. It is described in detail in Sections 4.4 and 5.3.

Here is a formalization of the introductory reasoning example given above:

	Cassie's beliefs	
1	$!\mathsf{Dog}(\mathsf{Fido})$	belief hypothesis
2	$!\forall \mathsf{x}\,(\mathsf{Dog}(\mathsf{x}) \Rightarrow \mathsf{Animal}(\mathsf{x}))$	belief hypothesis
	....	
3	$!\mathsf{Dog}(\mathsf{Fido}) \Rightarrow \mathsf{Animal}(\mathsf{Fido})$	universal elimination, x/Fido, line 2
4	$!\mathsf{Animal}(\mathsf{Fido})$	modus ponens, lines 1 and 3

The box surrounding the four sentences serves as an abstraction of part of Cassie's mental space. Its significance will become clear later. The sentence on line 1 represents Cassie's belief that Fido is a dog. The annotation on the right is not part of Cassie's mind at all; it just serves as an explanation to us. It says that in our example this belief is a *belief hypothesis*; i.e., it is one of Cassie's base beliefs that is not justified by any other beliefs. Base beliefs are those that were introduced into Cassie's mind in some extra-logical way such as, for example, perception, or by being told. The sentence on line 2 represents Cassie's belief that all dogs are animals. To represent this general rule we use a universally quantified sentence similar to what we would have used in standard first-order predicate calculus. That dogs are animals is also one of Cassie's base beliefs. Apart from serving as reference points in the explanations, the line numbers on the left can be viewed as time stamps. At a particular time only the sentence on that line and all sentences with smaller time stamps are considered to be part of Cassie's mental space. Thus, at time 2 Cassie only believes that Fido is a dog and that dogs are animals.

Now (at time 2) somebody asks Cassie whether Fido is an animal. Since at that time she does not have an explicit belief about it, she tries to find out by using reasoning or inference. The next two lines show how Cassie can infer the belief in question by using a proof method called *natural deduction*. The sentence on line 3 results from applying the rule of *universal elimination* to the sentence of line 2. Universal elimination eliminates the universal quantifier of a quantified sentence by substituting an actual individual constant for the variable; in our example, we substitute Fido for x. The result of such a substitution is often called an *instance* of the universal sentence. At that point in time Cassie has the new belief that if Fido is a dog then he is an animal. Since she already believes that Fido is a dog (line 1), she can now apply the rule of modus ponens (or implication elimination) to lines 1 and 3, thus arriving at the desired result that Fido is an animal in line 4.

The individual inference steps are carried out by Cassie's reasoning engine, which is the implementation of an automatic proof procedure for the deductive system $D_{\mathbf{SL}}$. After every inference step the result gets added to Cassie's mind as a new explicit belief. Thus, if the same question comes up another time, it can be answered directly without any reasoning. Of course, if the number of such inferred results becomes very large, this simple scheme might not be sufficient anymore. However, we will ignore such issues of scale — even though they are important — and assume that it is possible to handle them with some sophisticated belief caching and indexing scheme.

It should be pointed out that these inference rules and their use by the inference engine are the lowest level of Cassie's rationality. Cassie cannot reflect upon them; she just reasons that way, since that is the way she was built. This is not really a restriction, since it is perfectly possible to represent inference rules as special forms of actions, for example, by using the formalism developed by Kumar [1994], and then have Cassie reflect upon her own reasoning by carrying out these inference actions. However, while she carries out this form of meta-inference, she will still need a lowest, built-in level of reasoning to do so. This does not seem to be all that different from the way humans reason. Humans obviously can reflect upon their own reasoning, often at multiple levels, but at some point they also have to "bottom out" and resort to statements such as "well, because that is just plain obvious". At that point they just have a very strong sensation of truth or correctness of something, but they cannot describe any more why they have that sensation.

2.5 Avoiding Logical Omniscience

An important consequence of our belief representation and reasoning scheme is that Cassie is **not** logically omniscient. Logical omniscience is an unwanted side effect mostly of modal epistemic and doxastic logics, where agents are extremely idealized by modeling them as believing the complete, infinite logical consequence set of their base beliefs. For example, if a logically omniscient agent has the base beliefs P and $\mathsf{P} \Rightarrow \mathsf{Q}$, then it automatically also believes Q, and $\mathsf{P} \wedge \mathsf{Q}$, and $\mathsf{Q} \wedge \mathsf{Q} \wedge \mathsf{P}$, etc., i.e., everything that logically follows from these two base beliefs. This problem is rooted in the use of possible world semantics, which is the most popular semantic model for modal logics. A standard formulation is that some agent a believes some proposition p iff the denotation of the sentence $\mathsf{B}(a,p)$ is true, written $[\![\mathsf{B}(a,p)]\!] = t$, according to some model-theoretic interpretation. Here we used B as a modal belief operator. In possible world semantics the denotation of a modal sentence such as $\mathsf{B}(a,p)$ is usually defined in terms of the denotation of p by saying that $[\![\mathsf{B}(a,p)]\!] = t$ iff $[\![p]\!] = t$ in all worlds of a particular kind. Unfortunately, in standard treatments all worlds of the particular kind that make $[\![p]\!]$ true always also make all its logical consequences true; thus $[\![\mathsf{B}(a,q)]\!]$ is true for all of p's logical consequences q, which means, according to the chosen model, that the agent believes all those consequences.

In order for Cassie to believe that Fido is an animal the proposition term $\mathsf{Animal(Fido)}$ (or some other appropriate representation) has to be present in her mind, and its associated belief flag (the exclamation mark) has to be set, said differently, Cassie's mind has to contain the sentence $\mathsf{!Animal(Fido)}$. If either one of these conditions is violated, then she simply does not believe anything to that effect. This means that the notion of belief is tied to the makeup of the actual machinery that adds these propositional terms to Cassie's mind and sets and clears their belief flags. In the above example, only after the inference engine has gone through whatever "motions" are necessary to derive $\mathsf{!Animal(Fido)}$, and has added it to Cassie's mental space, does Cassie believe that Fido is an animal. Since such derivations actually do take time in an actual implementation, it is very appropriate to view the line numbers as time stamps as suggested above. Obviously, according to this belief model, Cassie is not logically omniscient. In a way, the notion of belief is really outside of our logic **SL**. **SL** can only prescribe what the legal inferences are, and how to make implicit information explicit. When and where Cassie applies these rules of inference to arrive at new beliefs is totally up to her and outside of the logic. This should become more clear once we present a more detailed semantic account in Section 4.3.

2.6 Multi-Contextual Reasoning

So far we assumed Cassie's mind or mental space to be one big, unstructured bag. Everything she could ever believe or consider had to be in there. However, there are situations where this simple scheme is not sufficient. One of them is *hypothetical reasoning*. For example, humans seem to be quite proficient in putting themselves into hypothetical situations such as "if I had won the lottery yesterday, I would have bought that nice house down the street,...", and think and reason in them as if they were actually the case. Planning is another common activity that involves hypothetical reasoning. While the reasoning in hypothetical situations is very much the same as in actual or real situations, some of the underlying base beliefs are not. They are only *hypothetical*, and most (sane) humans are very good at separating hypothetical beliefs from those concerned with reality. This separation is very important, since some hypothetical beliefs such as "I won the lottery" might outrightly contradict what in actuality is the case.

One possible account for the separation could be to view hypothetical beliefs as having a different epistemological status, for example, one could say that they are not actual beliefs but only *assumptions*. Translating that to our belief model we could use a different sentential operator instead of the '!', e.g., a '?', to mark those propositions that are only assumed but not really believed, and then throw them right into Cassie's mental bag. However, hypothetical reasoning requires a more complicated scheme than that. Hypothetical beliefs certainly do have a different status than real beliefs, but additionally, they *shadow* various real beliefs which in the hypothetical situation do not apply. For example, in the "if I had won the lottery" situation, all the direct consequences of actually not having won the lottery, e.g., not being able to pay this month's rent, should not have any impact on the reasoning in the hypothetical situation.

Our solution to this problem is to partition Cassie's mind into multiple reasoning *contexts*. Every

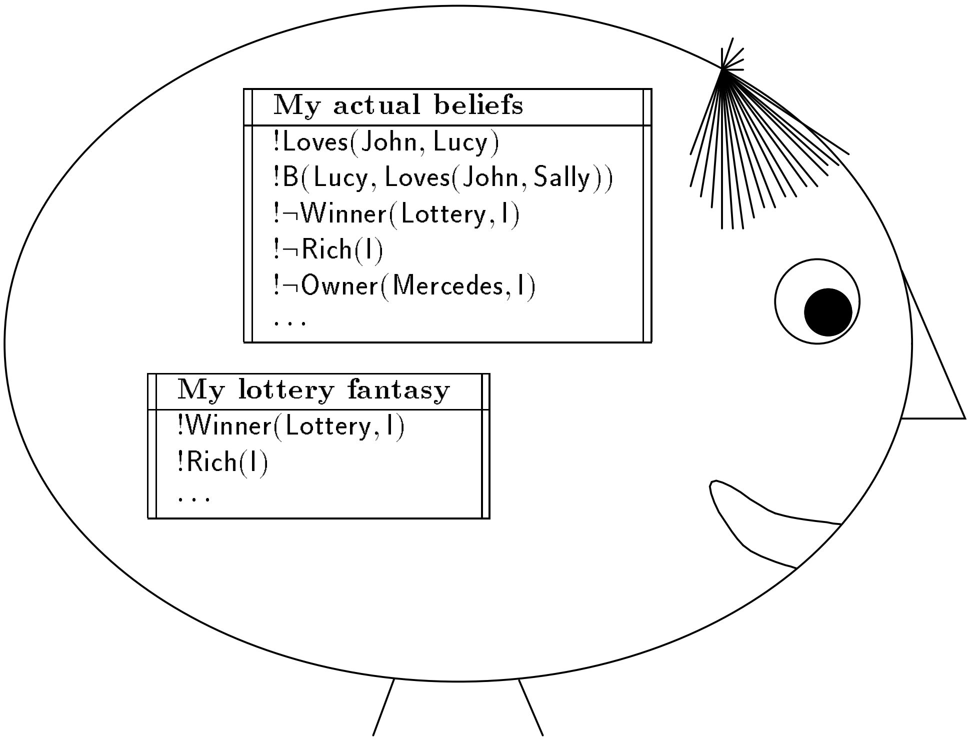


Figure 2.4: Cassie as a multi-contextual reasoner

such context is a mental space in its own right that contains a set of sentences which exclusively constitute Cassie's beliefs while she reasons in that context. Contexts are named, persistent, belief bases whose content can change over time. We usually draw them as boxes, as already done in the introductory reasoning example presented above. At any point in time there is a designated *current context* which is the one in which Cassie currently reasons. Figure 2.4 shows an example where Cassie's mind is partitioned into multiple contexts. Note that the lottery fantasy context contains beliefs which contradict Cassie's actual beliefs.

Our context model is derived from Martins's Multiple Belief Reasoner, or MBR, [Martins, 1983; Martins and Shapiro, 1988; Martins and Cravo, 1991]. MBR is an abstract specification of a context-based belief revision system. MBR operates on a knowledge base containing propositions associated with a support. These are also called *supported well-formed formulas*, or swffs. Supports are computed by the underlying logic. They record on what the derivation of a particular swff was based. In MBR a context is a set of hypotheses (similar to our belief hypotheses), and every operation on the knowledge base such as query, addition, etc., is associated with a context. The context associated with the current operation is called the *current context*. According to Martins [1988, p. 49] "a context determines a *belief space*, which is the set of all the hypotheses defining the context and all the propositions that were derived exclusively from them." This definition is slightly problematic, since according to it a context defines many belief spaces, one for every different set of propositions derived from it, but the definition reads as if every context would determine only one belief space. Since Martins does not want a belief space to be the deductive closure derivable from a context — in that case there would be only one — a better definition would be that *at any point in time* a context determines a *belief space*, which is the set of all the hypotheses defining the context and all the propositions that were derived exclusively from them *at that time*. All that aside, at any point in time the current context determines a belief space called the *current belief space*. A proposition is said to be *believed* if it belongs to the current belief space.

The contextual agent model described above is basically an application of MBR for the construction of a multi-contextual reasoning agent. The main difference from MBR is that our notion of context is *intensional* while MBR uses an *extensional* notion. In MBR a context is defined extensionally as a set of specific hypotheses, for example, $\{h_1, h_2, h_3\}$. Adding another hypothesis to such a set creates a new context. In our model, contexts persist even when new belief hypotheses get added to them; thus, a context is defined intensionally as a set of sentences associated with a particular context name, for example, $\{h|Ct(h) = Cassie\}$. The MBR-notion of context corresponds more closely to our notion of *belief state*. Belief states are defined and described in detail in Section 5.4.1.

Similar to Cassie's inability to reflect upon inference rules and how she uses them, she cannot reflect upon reasoning contexts. When contexts are created, and how they get used in inference, is decided by the underlying logic and the inference engine. This is a simplification that should be eliminated in future versions of SIMBA, since context can encode very important information. For example, while Cassie reasons within the lottery fantasy context, she should be aware that that is the context she is currently reasoning in, and that the propositions in it are not actual beliefs but merely fantasies. By "being aware" we mean that she should have beliefs to that effect, which means that her mind would need to contain sentences to represent such beliefs. In the current model this is not done. While Cassie reasons within the lottery fantasy context, she believes everything in it with the same conviction as when she reasons within her main context of actual beliefs. Moreover, what makes Cassie switch between contexts is determined completely by the inference engine, but not by her deliberate mental action. Cassie cannot make a choice about context switches, since in the current model she cannot represent such actions or reason about their utility and effects. Instead, whenever the inference engine demands, Cassie will be switched to a different state of mind without being able to explain that such a switch happened or why. The implementation of the inference engine does provide explanations for the application of inference rules and context switches (see Chapter 6), but these explanations are only the result of a lower-level tracing facility which is not actually part of Cassie's "mind".

Just as for inference rules, there must be a lowest level of context representation at which contextual awareness bottoms out, since belief is defined in terms of context; thus, beliefs about contexts have to – at least at some point – use a primitive, built-in notion of context. While the formalism developed by Kumar [1994] is a natural candidate to use for reflection upon inference rules, no immediate solution seems to be available for representing contexts and the reasoning *about* them. The formalization of context developed in [Guha, 1991; McCarthy, 1993] could serve as a possible starting point.

2.7 Simulative Reasoning

The main motivation for designing and building SIMBA is to give an agent such as Cassie the ability to reason with and about the beliefs of other agents. So far we only described various prerequisites necessary to attain that goal. The section describing the belief model showed how Cassie can hold beliefs, and the reasoning section described how she can reason with them, possibly in multiple contexts. The reason for choosing a belief representation language whose central representation mechanism are proposition-valued functions was that it allowed for a very natural representation of nested beliefs. What we have not yet shown is how to reason with these nested beliefs.

Cassie's beliefs about the beliefs of other agents are represented by sentences of the form $!\mathsf{B}(a,p)$; for example, the set of sentences in her mind of the form $!\mathsf{B}(\mathsf{Lucy},p)$ constitutes her beliefs about Lucy's beliefs. Similar to what was the case for Cassie's "plain" beliefs, not everything of her view of Lucy's beliefs is stored explicitly; thus, she has to use reasoning to uncover such implicit beliefs. For example, imagine Cassie and Lucy being engaged in a conversation where Lucy tells Cassie the following:

> "John is in love with Sally! Can you believe that? Sally, who changes men more often than other people change their shirts. Whoever falls in love with her really must be a fool."

A possible response by Cassie to the above statement is "I don't think John is a fool", maybe because she knows John as a very rational and intelligent person, maybe because she believes that he is actually in love with Lucy, or for whatever reason. If Cassie generally follows the Gricean maxim of relation or *relevance* in conversation [Grice, 1967], then her negative response must have been triggered by Lucy's direct or indirect statement that John is a fool, since that is what Cassie denies. But Lucy did not specifically say that John is a fool. However, if Lucy's utterance was felicitous, then she must believe that John is in love with Sally, and that everybody in love with Sally must be a fool; thus, in all likelihood, Lucy also believes that John is a fool, justifying Cassie's response. The crucial inference Cassie had to make before she could respond was that Lucy believes that John is a fool, notwithstanding that she herself does not think he is.

One way for Cassie to make this inference is *to think like Lucy*. Thinking like Lucy involves hypothetically assuming Lucy's beliefs, more precisely, what Cassie believes to be Lucy's beliefs, and then to reason with these beliefs the way Lucy would, more precisely, the way Cassie believes Lucy would reason with them. Figure 2.5 shows how that can be done using the multi-contextual reasoning approach motivated above. The reasoning context labeled "My beliefs" contains Cassie's actual beliefs relevant to our example discourse. The numbers on the left side of the context box serve as reference points or time stamps (similar to what we used in the introductory reasoning example). The numbers are not part of Cassie's beliefs. The sentence on line 1 represents Cassie's belief that John loves Lucy, and the one on line 2 represents her belief that John is not a fool. For

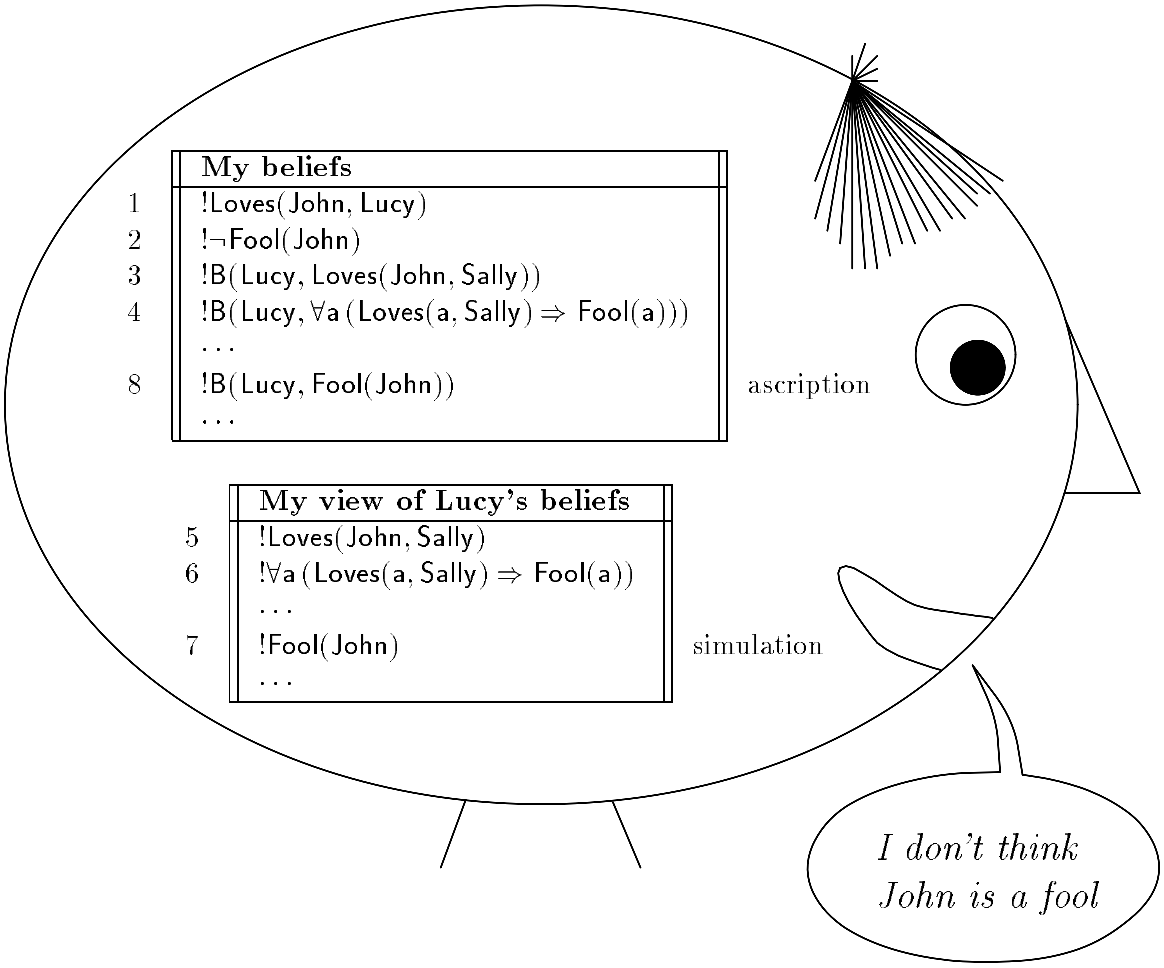


Figure 2.5: Cassie as a simulative belief reasoner

this example we will assume that Cassie already had these two beliefs prior to the conversation with Lucy. In contrast to Cassie's view on the matter, the sentence on line 3 represents Cassie's belief that Lucy believes that John loves Sally. Line 4 represents her belief that Lucy believes that everybody who is in love with Sally is a fool. We will assume that the latter two beliefs stem from Cassie's analysis or understanding of Lucy's utterance. Performing this analysis automatically by using natural language understanding techniques is far from trivial. We will, however, always assume that a proper representation of the content of such an utterance is already available to us, and not be concerned with the general issue of extracting such content automatically.

The reasoning context labeled "My view of Lucy's beliefs" (for short, "Lucy context") is used by Cassie to think like Lucy. As a first step, Cassie has to assume Lucy's beliefs as her own. This is done by unwrapping the object propositions p from their $\mathsf{B}(\mathsf{Lucy}, p)$ wrappers and importing them into the Lucy context. The results of this importation step are the sentences on lines 5 and 6. Thus, while Cassie reasons in the Lucy context, she (hypothetically) believes that John loves Sally and that everybody who loves Sally is a fool. Now Cassie has to reason the way she thinks Lucy reasons. Since in our model Cassie cannot reflect upon her own reasoning and inference rules, she also cannot reflect upon somebody else's. Hence, instead of Cassie reasoning the way she thinks

Lucy reasons, she simply reasons the way she always does. This means that Cassie attributes her own inference rules identically to other agents while she "thinks like them". The sentence on line 7 is the result of Cassie's reasoning from sentences 5 and 6 (the individual inference steps of universal elimination and modus ponens are omitted in the example). Line 7 is labeled "simulation", because its sentence was generated by Cassie's simulation of Lucy's reasoning. Cassie thinks like Lucy by *simulating her reasoning* according to the model she has of Lucy. This simulation is performed in a *simulation context*, which, in our example, is the Lucy context.

What is left to do is to make proper use of the simulation result. Since the simulation of Lucy's reasoning yielded the sentence on line 7, it is reasonable for Cassie to ascribe it to Lucy as one of her actual beliefs. This is done on line 8, which is labeled "ascription". Now even when Cassie reasons in her own actual reasoning context, she will believe that Lucy believes that John is a fool. This allows her to generate an appropriate response in the example conversation.

Simulative reasoning is an idea that has been discussed in the Artificial Intelligence literature for quite some time, for example, [Moore, 1973; Moore, 1977; Creary, 1979; Haas, 1986; Barnden, 1992; Chalupsky, 1993; Barnden *et al.*, 1994a; Barnden *et al.*, 1994b]. Its most appealing aspect is the economy inherent in it, in particular, when compared to hierarchical or meta-reasoning approaches. Simulative reasoning can be viewed as a simple *recursive call* to the underlying reasoning engine; thus, the reasoning about the reasoning of another agent is just as efficient as reasoning at the top level. In meta-reasoning approaches such as the one proposed by Konolige [1982], the reasoning of other agents is described at a meta-level; that is, a meta-theory is formed about the object theory that describes an agent. To reason about one inference step at the object level, several inference steps at the meta-level have to be performed. This inefficiency becomes even more dramatic once several levels of belief nesting have to be considered. Haas [1986, p. 262] gives a good characterization of the deficiency of meta-reasoning approaches of this kind. The meta-reasoning architecture described by [Aiello *et al.*, 1991] somewhat alleviates this problem by relying on the implementation of a *reflection* rule by the underlying FOL system [Weyhrauch, 1980]. The added efficiency, again, results from the use of *simulation structures* that allow the system to make inferences at the object (or base) level whenever possible, and then to lift the results of such object-level simulations up to the meta-level.

Simulative reasoning has a somewhat bad reputation, since it is often associated with the database approach to belief representation. According to that method, a situation such as "Lucy believes p" is represented by adding the sentence p to a belief database for Lucy's beliefs. For example, if we had used the database approach in the example in Figure 2.5, Cassie's beliefs about Lucy's beliefs would have been represented exclusively by the sentences in the Lucy context, and Cassie would not have had the associated $\mathsf{B}(\mathsf{Lucy}, p)$ sentences in her top-level context. Instead, she would have had some belief that the sentences in the Lucy context represent Lucy's beliefs. Applying simulative reasoning in its simplest form to such a belief database amounts to deriving new sentences from the set that is currently present and adding the derived results to the database.

Moore's criticism of this approach [Moore, 1977, p. 224] mainly focuses on its representational shortcomings. For example, the database approach does not accommodate for the representation of disjunctive belief such as "Lucy believes p or Lucy believes q", since there is no sentence that we can add to the Lucy database that would express that disjunction. Note, that adding the sentence $p \vee q$ would mean that "Lucy believes $p \vee q$". A possible solution for this problem would be to use two databases for Lucy's beliefs, one containing p and one containing q, and then to assert at the meta-level that Lucy either believes the sentences in database one or the ones in database two. However, with more such disjunctions the number of necessary databases grows exponentially, since we would need one database for every possible combination. A similar problem arises with the representation of non-belief such as “Lucy does not believe p”, since, again, there is no sentence that we could add to the Lucy database to express that belief, leaving as the only solution to use a new database to collect all of Lucy's non-beliefs.

Creary [1979, p. 180] discusses how the representational shortcomings of belief databases are really independent of their utility for simulative inference, and he sketches how they still could be used in conjunction with a more expressive belief representation scheme for the purpose of simulative reasoning. Our version of simulative reasoning is very much in line with the version imagined by Creary. We have a powerful belief representation scheme that allows for the representation of disjunctive or non-belief, and the deductive system of our logic **SL** defines how simulation contexts are to be created and used for the purpose of reasoning. The most common case in which Cassie simply collects a few propositions believed by some other agent in a simulation context and then reasons with them is handled just as efficiently as the simple simulation used with the database approach. Reasoning with disjunctive and non-belief is more expensive, since it usually requires the creation of hypothetical simulation contexts that are different from the “main” simulation context for an agent. However, since in our approach contexts are not used for representational purposes, there is no need to create an exponential number of them, but only as many as are necessary for a particular inference. The full power of this scheme will become clear in Section 4.4.

2.8 Belief Ascription is Defeasible

A very important characteristic of belief ascription is that its results are *defeasible*. That is, regardless of what inference method is used, it is impossible to prove what some real, non-idealized agent must believe according to the beliefs one knows or assumes that agent holds. The most Cassie can do is to derive what another agent *should* believe according to her model of the agent and her own reasoning rules, but whether the other agent *actually* believes it cannot be determined by reasoning alone.

A belief ascribed to some agent based on simulating its reasoning can at the most be an educated guess that might be close to reality, but that could also be completely wrong, as everyone who has ever taught a class will know. For that matter, if teachers could accurately simulate the reasoning of their students, they would not have to give any exams. Even if Cassie had a completely accurate model of some agent's base beliefs, and her reasoning skills matched the ones of the other agent exactly, a result derived by Cassie during a simulation of that agent might still not be believed by it, if only for the fact that the other agent has not yet performed the necessary reasoning itself to arrive at the particular conclusion. What we are assuming here is that reasoning takes time, and that agents do not believe the infinite logical consequence set of their base beliefs. For example, suppose Cassie and her very close friend Oscar are both students who study complexity theory. While Cassie is working on some homework assignment she might prove a particular theorem, and thus come to believe it. If we assume that she believes that all the necessary premises are believed by Oscar also, and that Oscar is just as good as she is in proving theorems, then a simulation of Oscar's reasoning might lead her to believe that he also believes the theorem in question. However, Oscar might not have worked on the homework yet; hence, it is quite possible that he does not yet believe the theorem, since he has not carried out the proof himself. One could argue that in this example Cassie was not justified in carrying out the simulation, since she did not know whether Oscar had already worked on the homework or not. But there are many possible scenarios in which she would have been justified to assume that he had done the homework already, when in fact he had not.

An immediate consequence of this defeasibility is that any belief ascription mechanism has to address issues of belief maintenance and revision. Conclusions drawn by way of simulation might be contradicted by beliefs acquired more directly. In our example above, Oscar could have told Cassie about some difficulties he had in proving some other theorem which would have been very easy to prove with the first theorem available. From that Cassie might form a belief that Oscar does not believe the first theorem, which would directly contradict the result from her simulation of his reasoning. In such a situation, Cassie must know that the contradicting belief has been derived via simulation and can hence be withdrawn without harm in light of the new, stronger evidence to the contrary.

Here are a number of reasons why some belief $\mathsf{B}(a,p)$ derived by Cassie via simulation of some agent a's reasoning could be incorrect or contradict some already existing belief in Cassie's model of a:

1. Cassie's model of a is incorrect, because she made a mistake somewhere during the construction of the model, or because a lied.
2. Cassie's model of a is correct and the simulation is, too, but agent a has not yet done the necessary reasoning to have come to believe p.
3. Cassie's model of a is correct and inconsistent, because a is inconsistent.

How are these situations different from analogous ones occurring during Cassie's own top-level reasoning? The first case is certainly very similar to the situation in which Cassie's own set of beliefs is inconsistent. The second case, however, cannot occur for Cassie's own beliefs. The third case is a special case of the second, if we assume that whenever agents notice a direct inconsistency they remedy it. Under this assumption, the only way that Cassie's model of a can be correct yet inconsistent, is that a has not yet done the necessary reasoning to discover the inconsistency.

A reasoning mechanism that ascribes beliefs to other agents by way of simulation has to be able to deal with these various situations either by revising the set of beliefs that constitute an agent model or by blocking certain inferences. For example, it should be able to handle this set of belief sentences

$$!\mathsf{B}(a,p)$$
$$!\mathsf{B}(a,p \Rightarrow q)$$
$$!\neg\mathsf{B}(a,q)$$

without generating a contradiction, even though by simulation of a the belief $\mathsf{B}(a,q)$ would follow. As described in the example above, a might not yet have inferred q. Note, that $!\neg\mathsf{B}(a,q)$ means that Cassie believes that a does not believe q, but not that Cassie does not believe that a believes q. The only way she can have a belief about not holding a particular belief herself, is by means of negative introspection, e.g., $!\neg\mathsf{B}(\mathsf{I},\mathsf{B}(a,q))$. However, the special inference rules of the deductive system of **SL** that govern introspection will allow her to derive $!\neg\mathsf{B}(\mathsf{I},\mathsf{B}(a,q))$ whenever she believes $!\neg\mathsf{B}(a,q)$ and vice versa.

SIMBA uses the following strategies to cope with the defeasibility of simulation results: (1) They are treated as *default* conclusions; thus, they can be shadowed by beliefs that were acquired more directly. (2) The inference rules of the logic **SL** compute proper support sets for the application of every inference rule; thus, the logic has a belief maintenance system built right into it. Removing (or "disbelieving") a certain belief hypothesis as the result of belief revision will automatically remove (or disbelieve) all sentences that depended on it.

SIMBA also has a notion of *belief entailment*, which allows Cassie to explicitly reason about a situation where the beliefs of some other agent entail a certain simulation result, even though the simulated agent does not believe what its beliefs entail. The full benefit of this concept will become clear in Section 5.5.

2.9 Group Belief

In the introductory simulative reasoning example presented in Section 2.7, Cassie only had to represent the beliefs of Lucy, and all of Cassie's beliefs about Lucy's beliefs were represented explicitly by propositions of the form $\mathsf{B}(\mathsf{Lucy}, p)$. According to our logic **SL**, all of Cassie's beliefs are completely private, and the only way a proposition can become available to take part in the simulation of an agent such as Lucy is if it is wrapped inside an appropriate belief proposition. In many cases, it will be sufficient to represent the beliefs of another agent explicitly one-by-one; however, in some cases it might be desirable to tell Cassie about typical beliefs of certain groups or classes of agents. For example, we might want to tell her that all theoretical computer scientists believe that the Halting Problem is undecidable. Since in **SL** we can quantify over agents, this could be done in the following way:

$$!\forall \mathsf{a}\ \mathsf{ThCompScientist}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \mathsf{Undecidable}(\mathsf{HP}))$$

Then whenever during Cassie's simulation of some agent's reasoning it becomes necessary for her to determine whether that agent believes the Halting Problem to be undecidable, all she needs to do is to check whether the agent is a theoretical computer scientist. This extra reasoning where Cassie has to go outside of the simulation context to derive a particular belief proposition of the simulated agent in a meta-context before it can be used in the simulation is called *meta-reasoning*.

In a similar vein, we could tell Cassie about a *common belief* such as "everybody believes that dogs are animals" in the following way:

$$!\forall \mathsf{a}\ \mathsf{Agent}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \forall \mathsf{x}\ \mathsf{Dog}(\mathsf{x}) \Rightarrow \mathsf{Animal}(\mathsf{x}))$$

Both of these representations of group belief go only one level deep. Thus, they allow Cassie to make use of such a belief in the simulation of a qualifying agent, but in a simulation carried out at a deeper level of nesting; e.g., if Cassie simulates the reasoning of one theoretical computer scientist who in turn simulates that of another, the group belief would not be available any more at the second level of nesting, unless it is represented as a belief of the first computer scientist. For example, we could represent that Cassie believes that Oscar knows about the group belief with the following sentence:

$$!\mathsf{B}(\mathsf{Oscar}, \forall \mathsf{a}\ \mathsf{ThCompScientist}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \mathsf{Undecidable}(\mathsf{HP})))$$

Even more generally, we could tell Cassie that all theoretical computer scientists have a belief about this group belief:

$$!\forall \mathsf{a}\ \mathsf{ThCompScientist}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \mathsf{Undecidable}(\mathsf{HP}))$$
$$!\forall \mathsf{a}\ \mathsf{ThCompScientist}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \forall \mathsf{b}\ \mathsf{ThCompScientist}(\mathsf{b}) \Rightarrow \mathsf{B}(\mathsf{b}, \mathsf{Undecidable}(\mathsf{HP})))$$

With this last set of sentences, the particular belief held by theoretical computer scientists could be used in all simulations that involved at most two levels of nesting.

Often, however, common or group belief is viewed as something everybody (in the group) believes, and everybody believes that everybody believes it, and everybody believes that everybody believes that everybody believes it, etc. Since, in **SL**, propositions are ordinary terms of the language that can be quantified over, we can tell Cassie about such unrestricted group belief in the following manner: Suppose the function $\mathsf{CB}(p)$ represents the proposition that p is a common belief. Then we can tell Cassie how to make use of such a belief in arbitrarily nested simulations with the following sentences:

$$!\forall \mathsf{a}, \mathsf{p}\,(\mathsf{Agent}(\mathsf{a}) \wedge \mathsf{CB}(\mathsf{p})) \Rightarrow (\mathsf{B}(\mathsf{a}, \mathsf{p}) \wedge \mathsf{B}(\mathsf{a}, \mathsf{CB}(\mathsf{p})))$$
$$!\mathsf{CB}(\forall \mathsf{a}, \mathsf{p}\,(\mathsf{Agent}(\mathsf{a}) \wedge \mathsf{CB}(\mathsf{p})) \Rightarrow (\mathsf{B}(\mathsf{a}, \mathsf{p}) \wedge \mathsf{B}(\mathsf{a}, \mathsf{CB}(\mathsf{p}))))$$

The first sentence allows Cassie to derive every commonly believed proposition as a particular belief of an agent, as well as that the agent believes that it is a common belief. The second sentence defines the first one as a common belief, thus allowing both sentences to be imported into arbitrarily deeply nested simulation contexts.

The examples above demonstrate that in the design of the logic **SL** it is sufficient to only consider specific belief propositions $\mathsf{B}(a, p)$, since common and group belief can be dealt with by giving Cassie special domain rules that tell her how to translate the group belief into the specific belief propositions necessary for simulative reasoning. For the sake of efficiency, however, cases such as the unrestricted group belief are handled better with special inference rules. These can then be exploited by the implementation of the reasoning engine to efficiently make such propositions available in various reasoning contexts, without having to rely on their derivation by the general inference mechanism.

2.10 Limited Reasoning

A possible criticism of our direct simulation approach in which Cassie attributes her own reasoning skills *identically* to other agents during simulation, is that it would not allow her to accommodate for agents with limited reasoning abilities. For example, one might suggest that some of the inference rules used by Cassie, e.g., modus ponens, might not be known by some less intelligent agent whose reasoning she simulates. Thus, if she knows about such a deficiency of another agent she should be able to take that into account and not use that inference rule during the simulation in order to make its result more accurate.

We are strongly convinced, however, that limiting the application of particular inference rules would be the wrong strategy to adapt to limited reasoning skills of other agents. The inference rules of the logic **SL** build the fundamental substrate of Cassie's reasoning skills. Deactivating any one or more of them would have – to stay within the metaphor – a debilitating effect. For example, Cassie's reasoning without modus ponens would be very much impaired, similar to the way the reasoning of a human agent with a severe brain disorder would be impaired (the comparison is about the level of reduction, not about the absolute level of skill).

A much better way of accommodating to different abilities of various agents is by attributing particular expertise to some but not to others. For example, the group belief that theoretical computer scientists believe that the Halting Problem is undecidable as described in the previous section can be used by Cassie whenever she simulates such an agent, but it will not be used when she thinks about a "normal" agent. The "normal" agent might still have the same reasoning skills as the theoretical computer scientist; it just does not have the same expertise according to Cassie.

Another way to model limited reasoning abilities of other agents that is sometimes suggested is to limit the number of reasoning steps used or the amount of resources expended. Such cut-offs seem to be rather arbitrary, however, and their use demonstrates that this whole area is still not very well understood. An interesting question is also how Cassie could accommodate for agents who she believes are more intelligent than she is, since in her simulation of such an agent she can certainly not reason more effectively than she usually does in order to compensate for the perceived difference in ability.

2.11 Summary of Desiderata

Here is a summary of the various desiderata that guided the development of SIMBA and its underlying logic **SL**:

Representational Adequacy: The language of **SL** has to be a proper belief representation language that can be used as the language of thought for a computational agent such as Cassie. It must allow the proper representation of Cassie's own beliefs, as well as her beliefs about other agents at arbitrary levels of nesting. It must also be powerful enough to deal with various standard belief representation problems such as non-belief, introspection, disjunctive belief, *de re* and *de dicto* belief reports, quantifying into belief contexts, etc.

Proper Semantics: The language of **SL** should have a proper semantics that adequately reflects what it means for an agent to believe something, as well as what the objects of an agent's beliefs are viewed to be. Having the semantics reflect common intuitions about the nature of belief as a propositional attitude is also desirable.

Subjectivity: The logic **SL** should be subjective in the sense that it should specify how to *build* an actual agent with it as the foundation, as opposed to aid in the *analysis* of agents from an objective, outside observer's point of view.

Integration with Top-level Reasoning: The top-level reasoning of an agent and its reasoning about the beliefs of other agents should be seamlessly integrated. For example, a cognitively adequate model should not make it necessary to switch languages in order to reason about other agents (e.g., as necessary in hierarchical object-language/meta-language architectures). Results of the agent's own top-level reasoning should smoothly influence reasoning in simulation contexts and vice versa whenever appropriate.

Accurate Agent Models: The attribution of beliefs to other agents should be as accurate as possible. No beliefs held by Cassie or any other agent she has beliefs about should be accidentally attributed to a particular agent, unless that attribution is explicitly warranted.

Automatic Generation of Agent Models: It should not be necessary to fully explicitly specify the set of base beliefs held by a particular agent (also called an agent model). Instead, it should be possible for Cassie to automatically derive such beliefs whenever possible before they get used in the simulation of an agent. This is particularly important to allow the representation and use of group beliefs, i.e., beliefs that are commonly held by a particular group or class of agents.

Simulation at Arbitrary Levels of Nesting: Just as the language of **SL** must allow for the representation of arbitrarily nested beliefs, it must be possible to use such beliefs in simulations at arbitrary levels of nesting. For example, Cassie must be able to simulate Lucy's simulation of Sally's simulation of John, or Lucy's simulation of John's simulation of herself, Cassie.

Defeasibility of Simulation Results: Perhaps our most important objective is to handle the defeasibility inherent in belief ascription. Simulation results can at best be seen as plausible *assumptions* that might have to be withdrawn in the light of stronger evidence to the contrary. The logic has to handle contradictions that arise due to simulations without jeopardizing Cassie's own top-level reasoning. It should also support belief revision and maintenance in case obsolete beliefs have to be withdrawn or revised.

Belief vs. Belief Entailment: Cassie should be able to explicitly reason about the incompleteness of the reasoning of other agents. Thus, she should be able to represent and reason about situations where the beliefs of an agent entail a certain conclusion, while at the same time that agent does not believe the conclusion in question.

Implementation: Finally, it is most important to provide an implementation of SIMBA, since many of its strengths and, in particular, weaknesses can only be found by experimentation with an actual system.

Chapter 3

Background

In this chapter, we review other work relevant to the problem we set out to solve. The relevant literature is vast, since, at least in principle, every logic-like formalism that deals with the representation of nested propositional attitudes, and which maybe also defines a proof procedure of some sort, is a candidate solution. However, as the discussions of various approaches will reveal, none of them seems to be fully adequate with respect to all of the desiderata stated previously. For the most part, our focus will be on the review of approaches that have actual implementations associated with them, since we view such implementations as very important.

Formal treatments of propositional attitudes are quite intricate; hence, we will try to provide sufficient detail in the description of every formalism to give the reader an idea how certain problems are solved and how others are not. In general, we will concentrate on the reasoning aspects of a certain approach; however, in some cases, the representational approach to belief will be criticized as well. Depending on the work under discussion, the notation will vary to provide as close a connection to the original publication as possible.

3.1 Moore's Reasoning about Knowledge and Action

Moore [1980; 1985] develops a formalism that aims at representing the concepts of *knowledge* and *action*, together with their strong dependencies, exhibited by certain problems in the domain of planning and acting. Moore motivates the important role of knowledge in a planning situation with the example of an agent who wants to open a safe. Among the preconditions for such an action are not only the physical ability of the agent to dial the combination, but, most essentially, the agent's *knowledge* of the combination that will open the safe.

The propositional part of the modal logic of knowledge used by Moore is described by the following axiom schemata in which P and Q are variables ranging over propositions, and A is a variable ranging over terms denoting agents (this axiomatization is taken from [Moore, 1985], which contains a condensed and updated version of the work described in [Moore, 1980]):

M1. P, such that P is an axiom of propositional logic
M2. KNOW(A,P) $\supset$ P
M3. KNOW(A,P) $\supset$ KNOW(A,KNOW(A,P))
M4. KNOW(A, (P $\supset$ Q)) $\supset$ (KNOW(A,P) $\supset$ KNOW(A,Q))
M5. If P is an axiom, then KNOW(A,P) is an axiom.

Every such axiom schema defines an infinite set of actual axioms which, in conjunction with the inference rule modus ponens, define the theorems of the system. A similar axiomatization for the characterization of knowledge is due to Hintikka [1962].

Because of difficulties in using this logic directly in a computational system, Moore gives a Kripke-style possible-world analysis [Kripke, 1963] of knowledge which can be formalized within ordinary first-order logic, and hence permits the use of standard automatic theorem proving techniques to perform reasoning about knowledge. A statement of the form KNOW(A,P) is analyzed by the introduction of an *accessibility relation* K, such that $K(A,W_1,W_2)$ means that the possible world (or state of affairs) W_2 is compatible or consistent with what A knows in the possible world W_1. For example, if A knows P in the actual world W_0, then P must be true in every possible world W_1 for which $K(A,W_0,W_1)$ holds. Analyzing the various axioms of the modal logic of knowledge imposes various constraints on the relation K; for example, the analysis of axiom M2 implies that K for a given knower has to be reflexive, while the analysis of M3 implies that it also has to be transitive. Using the above analysis, an assertion of the form KNOW(A,P) can be replaced by an ordinary first-order formula of the form

$$\forall w_1(K(A,W_0,w_1) \supset T(w_1,P))$$

in which possible worlds are treated as individuals that can be quantified over. The operator T expresses truth relative to a possible world; i.e., T(W,P) means P is true in possible world W. This operator is extensional and distributes over ordinary logical connectives, which is not true for the modal operator KNOW. The above translation of A knows P in the actual world W_0 is that in all worlds accessible for A via the relation K, P must be true. Similar to this translation, Moore develops a complete first-order formalization of the modal logic of knowledge, i.e., a first-order *meta-language* that denotes formulas of the modal *object language*. As a proof system, he describes a natural deduction system that can be used to perform reasoning about knowledge and action. Moore [1980] also gives a simple computational model for this natural deduction system, which is used in the hand-simulation of some "automatic" deductions that solve various problems involving reasoning about knowledge and action.

Removing M2 from the above axiomatization would give a modal system for *belief*; i.e., believing a certain proposition would no longer entail that proposition to be true (truth is usually seen as a necessary ingredient for *knowledge*, thus, in order for an agent to know something, in this technical sense of knowledge, that something — among other things — has to be true). Moore, however, does not want to deal with belief, because of the complications in the formal treatment of issues such as non-monotonicity and belief revision necessary for a belief system. In particular, by using knowledge rather than belief for the model of an agent's reasoning, the agent does also know the effects of its actions. Since whatever the agent *knows* must be true, the agent can always be sure that its actions are effective (it cannot know that an action has a certain effect, if the occurrence of this effect is not guaranteed). This enables a relatively simple formalization which would have to be much more complicated for a system in which actions could sometimes be non-effective.

Discussion

While the possible-world analysis of the modal logic of knowledge enables an elegant formalization of an agent's knowledge and his reasoning about knowledge and action, it constitutes a gross oversimplification that leads to a very unrealistic model for cognitive agents: First, the possible-world analysis described above implies that agents are ideal (or logically omniscient) reasoners; i.e., apart from the propositions that they explicitly know, they also know all the logical consequences of these propositions (suppose A knows P and P$\supset$Q, i.e., P and P$\supset$Q are true in all worlds compatible with what A knows; hence, Q must also be true in all these worlds, which means that A knows Q). Logical omniscience is certainly not a property of realistic cognitive agents.

The other unrealistic component of Moore's theory stems from the usage of knowledge rather than belief, since that implies that everything an agent believes is actually true. In the context of agent models and reasoning about other agents' knowledge, this means that agent models are completely accurate and that the other agents, too, are assumed to be ideal reasoners. While Moore's first-order characterization of the modal logic of knowledge in principle allows automatic reasoning about other agents' knowledge, its idealized assumptions about agents' beliefs and reasoning abilities prohibit its use as a model of a realistic cognitive agent.

Moore's logic in conjunction with an automatic theorem prover has been used as a part of the KAMP natural language generation system. Appelt [1985, p. 158], however, states that the possible worlds formalization of the **Know** operator leads to a very large number of irrelevant inferences.

3.2 Creary's Fregean Representations and Simulative Reasoning

Creary [1979] develops a Fregean representation scheme for the representation of information about propositional attitudes. This representation scheme contains a notational system for a potentially infinite, multi-branched hierarchy of concepts, as well as a special name-forming machinery to provide for names denoting quantified propositions and "definite description" concepts of individuals. With this machinery, Creary is able to represent various different readings of sentences that describe iterated (or nested) propositional attitudes, as, for example, expressed in the sentence: "Pat believes that Mike wants to meet Jim's wife." For the representations of three different readings of this sentence, the reader is referred to [Creary, 1979].

What is relevant for us here is the second part of Creary's paper, in which he describes how his representational scheme could be used to reason about other organisms' (as he calls them) propositional attitudes by way of *simulative reasoning*. He characterizes his approach as one in which the program that reasons about the propositional attitudes of others would use its own "mental machinery" to simulate the thinking of these other organisms. The simulation is imagined to be facilitated by a CONNIVER-like context mechanism [McDermott and Sussman, 1972] in the following way: Suppose the program wants to know whether Mike believes that Jim is Sally's uncle. To do, that it tries to infer one of the following formulas

believes(mike, Uncle{Jim, Sally})
believes(mike, Not Uncle{Jim, Sally})

by setting up two contexts C_{bm} and $C_{\neg bm}$, the former containing formulas that are assumed to be believed by Mike, and the latter containing formulas assumed not to be believed by Mike. Both contexts are to be constructed from explicit information about Mike's beliefs as well as from a corpus of common beliefs. Then a knowledge-based theorem prover of the kind envisioned in [Nilsson, 1977] is supposed to prove one of the above formulas in the context C_{bm}, using the formulas in Mike's non-belief context as goals in order to derive a contradiction. Measures of the resources expended by the inference process are to be computed for use in conjunction with knowledge about the level of Mike's reasoning abilities. The contexts are supposed to be kept fairly small. They only get extended if absolutely necessary taking into account the relevance of newly introduced propositions. Finally, the simulative process must be callable recursively in order to permit reasoning about iterated propositional attitudes.

Discussion

Creary's outline of a simulative reasoning mechanism describes many features that are also desired for SIMBA. Among these are the attribution of an agent's own mental abilities to other agents in order to simulate their reasoning, usage of a context mechanism in order to carry out the simulation using only the beliefs relevant to these agents, as well as the ability to perform simulations at arbitrary nestings. However, as the various quotations above indicate, Creary only gives a general outline of how a simulative reasoning mechanism could operate; there is no real system that actually deals with the various complicated problems such as determining the beliefs held by an agent (in the context formation phase), finding a proof for a certain proposition, "judicious choice of relevant propositions" [Creary, 1979, p. 179] to be used in the simulation, etc. Creary does not say anything about the defeasibility of the simulation process, even though he mentions that "raw results of the simulation can be corrected on the basis of known differences between the simulator and Mike (the simulated agent)" (p. 180). Creary's analysis serves as a good starting point for the development of a system such as SIMBA. Unfortunately, he never developed an actual prototype as a follow-up of his initial design.

3.3 Maida's Reasoning about Knowing

Maida [1983] describes a representation formalism based on intensional individuals that can be used to represent iterated propositional attitudes. He applies the formalism to represent a variation of McCarthy's famous telephone number problem [McCarthy, 1979] and uses *indirect simulation* of another agent's reasoning to conclude what some agent must know assuming he knows the premises of the problem. Quite similar to Creary, Maida motivates this form of reasoning by what he calls the

> Axiom of Rationality: If a cognitive agent knows or is capable of deducing all of the premises of a valid inference, then he is capable of deducing the conclusion of that inference.

While the representation formalism described is a necessary prerequisite for being able to do simulative reasoning, it is not sufficient, and Maida fails to give a computational account for how the reasoning could be done.

In [Maida, 1986], he goes further by demonstrating how an introspective agent can use analogy-based reasoning to construct an architecture for reasoning about other agents' beliefs based on introspective examination of its own architecture, and how by that, for example, it can realize that the other agent can reason by modus ponens. The main purpose of this exercise is to demonstrate how by using only introspection and a certain form of analogical reasoning, a belief space architecture (a tree of nested belief spaces) could be generated automatically by an agent, without assuming that this architecture is a defining ingredient of the agent. Again, the exercise is only theoretical, and no precise computational account is given of how this could be used in an actual belief reasoner.

3.4 Konolige's Deduction Model of Belief

The *deduction model of belief* was developed by Konolige [1986]. The central concept of this model is that of a *belief system*, which is an abstraction of the part of an agent's cognitive makeup responsible for beliefs. A belief system is formalized as a *deduction structure* $\langle B, \mathcal{R} \rangle$, where B is a set of sentences in some logical language L, the language of belief, and $\mathcal{R}$ is the set of deduction rules of the belief system. The belief set $\mathbf{bel}(\langle B, \mathcal{R} \rangle)$ of a deduction structure $\langle B, \mathcal{R} \rangle$ is the base set B, plus all the sentences derived using the rules $\mathcal{R}$, or formally:

$$\mathbf{bel}(\langle B, \mathcal{R} \rangle) =_{def} \{\Phi | B \vdash_{\mathcal{R}} \Phi\}$$

Note that the deduction rules $\mathcal{R}$ could be incomplete or could impose a cost bound on deductions, which, at least in theory, allows for a very accurate modeling of an agent's reasoning abilities. It is also assumed that a belief system described by a deduction structure is *deductively closed*; i.e., anything derivable from the base set of beliefs can participate in further derivations. This essentially means that an agent's belief derivation process cannot distinguish between derived beliefs and base-set beliefs.

Konolige develops two modal languages $L^{\mathbf{B}}$ and $L^{\mathbf{Bq}}$ that can serve as languages for belief. Both are built on top of a standard first-order language L_0 without function symbols. In $L^{\mathbf{B}}$, the modal belief operators can only take sentences as arguments, whereas in $L^{\mathbf{Bq}}$ they can also take sentence schemas, which makes it possible to quantify into belief contexts for greater expressiveness. In both languages, belief is represented by a modal construct of the form $[S_i]\Phi$. Φ is a sentence in the internal language of the agent denoted by the term $[S_i]$, which in turn could be $L^{\mathbf{B}}$ or $L^{\mathbf{Bq}}$ to allow the representation of nested beliefs. The intended meaning of the construct is that agent S_i believes the sentence Φ. The sentence $[S_i]\Phi$ is interpreted as being true iff Φ is in S_i's belief set; i.e., the meaning of belief sentences is defined in terms of deductively closed belief sets.

The interpretations for the languages $L^{\mathbf{B}}$ and $L^{\mathbf{Bq}}$ are given as first-order model structures which are augmented by a set of deduction structures D, one for every agent S_i. The relation between the satisfiability of sentences containing belief operators in the external language, and derivations in the internal language is stated by the following lemma (for $L^{\mathbf{B}}$):

Attachment Lemma: The (denumerable) set $\{[S_i]\Gamma, \neg[S_i]\Delta\}$ is unsatisfiable iff for some $\delta \in \Delta, \Gamma \vdash_{\rho(i)} \delta$ (where $\rho(i)$ is the set of deduction rules of agent S_i).

What is important for us here is the way in which the deduction model defines a formal proof procedure that is automatable and that can be used to derive new beliefs of agents given a set of base beliefs held by these agents. In [Konolige, 1986], he describes a resolution-based proof system R for $L^{\mathbf{Bq}}$. In addition to Robinson's binary resolution rule [Robinson, 1965], this system contains the B-resolution rule to deal with belief literals. B-resolution is a special case of *total narrow theory resolution* developed by Stickel [Stickel, 1985]. Here is its definition:

$$\begin{array}{ll}
[S_i]\Phi_1 & \vee A_1 \\
[S_i]\Phi_2 & \vee A_2 \\
\vdots & \\
[S_i]\Phi_n & \vee A_n \\
\neg[S_i]\psi & \vee A \\
\hline
A_1 \vee A_2 \vee \cdots & \vee A_n \vee A,
\end{array} \quad when\ \Phi_1^\bullet, \ldots, \Phi_n^\bullet \vdash_{\rho(i)} \psi^\bullet$$

The rule is justified by the attachment lemma; it basically says that if we can deduce ψ from premises Φ_i using agent S_i's deduction rules, even though the agent does not believe ψ, then these beliefs are unsatisfiable, and, hence, we can conclude the disjunction of the A_i's. The bullets in the deducibility condition of B-resolution indicate a naming map translation that deals with the problem that an agent might use different constants in its internal language to refer to objects in the domain than the constants used in the external language outside of the belief operator. The technical problems associated with these naming issues are described in detail in [Konolige, 1986]. They do not concern us here, since they are completely avoided by the subjectivity of our logic **SL**.

Geissler and Konolige [1986b] describe a resolution algorithm for the quantified modal (belief) logic employed by the deduction model. The basic idea is to use a data structure called a *view* for every agent to perform the necessary deduction using the agent's beliefs and deduction rules. Assuming appropriately powerful deduction rules for the agent, the proof in the view can be found by a recursive call to the theorem prover. Because there is no decision procedure for unsatisfiability in quantified modal logic, the work done in the top-level resolution proof as well as in the various proofs performed in the views of agents has to be interspersed (or dovetailed) appropriately in order to find a proof if it exists. Basically, the theorem prover "time-shares" its attention between the various invocations of the proof procedure. An implementation of this resolution algorithm and its application to a famous benchmark problem for nested belief reasoning called *the Wise Man puzzle* is described in [Geissler and Konolige, 1986a] (see page 154 for a full description of the Wise Man puzzle).

Discussion

One of the main motivations for Konolige's model is to overcome the unrealistic property of logical omniscience exhibited by possible-worlds treatments of knowledge and belief such as Hintikka's or Moore's. In his model, an agent is modeled to believe everything that deductively follows from the agent's base set of belief by use of the (possibly logically incomplete) deduction rules of the agent; i.e., an agent is *deductively* omniscient instead of logically omniscient, which does not seem to be too much of an improvement. Once an agent's deduction rules are assumed to be saturated (i.e., logically sound and complete), Konolige can prove a correspondence theorem between deduction models and possible-world models for belief, which shows their very close connection.

The biggest shortcoming of the deduction model is that it does not at all deal with any aspects of defeasibility and belief maintenance; so, a belief sentence derived as a result of the application of the B-resolution rule is not distinguished in any way from a belief sentence given as an initial assumption. In particular, note that the formulation of the B-resolution rule implies that this model cannot contain the damage if any agent's beliefs are incomplete or inconsistent. For example, the negation of the last belief literal could not just be the negation of a sentence that one wants to prove with resolution refutation, but instead the statement of a particular non-belief of agent S_i. In that case the resulting set of belief literals will always be unsatisfiable and, hence, will allow the derivation of any sentence whatsoever from any set that contain them as a subset.

Suppose that the top-level formulas are the world model of Cassie, and all the belief sentences are Cassie's model of other agents she knows about. Then in a case such as the above, where the beliefs of some agent are incomplete, her reasoning would go completely haywire, since her set of beliefs is now inconsistent based on the incompleteness of an agent she knows about. To make things worse, the dovetailing nature of the proof procedure of Geissler and Konolige would insure that the inconsistency of a particular agent model would always come to bear even if that agent is not at all relevant in a particular reasoning task. To be fair, one must say that in [Konolige, 1986, p. 4] Konolige explicitly mentions belief revision and maintenance as issues not addressed at all by his model.

3.5 Haas's Syntactic Theory of Belief and Action

Haas [1986] describes a syntactic first-order theory of belief and action. In syntactic theories, *believe* is not a special predicate or modal operator, but rather an ordinary predicate that takes the *name* of a sentence as an argument instead of the sentence itself. Such a name is a quoted string of some sort that uniquely denotes an actual sentence. For example, in Haas's theory, the sentence "John believes that Lucy's phone number is 5766" would be expressed as

(believe John ('= ('PhoneNumber 'Lucy) '5766))

where the term ('= ('PhoneNumber 'Lucy) '5766) is meant to denote the sentence (= (PhoneNumber Lucy) 5766). The reason for explicitly quoting every single object instead of just using double quotes around the whole sentence is that this allows the use of variables at certain positions to create what Haas calls wff schemas.[1] These names for sentences or wffs are basically constants and would be treated as such by any first-order proof procedure. In order to be able to reason with the actual sentences expressed by such names (as necessary, for example, to find out whether an agent's beliefs entail a certain conclusion), Haas introduces the *reflection schema*, an infinite set of axioms each of which describe a proof or a class of proofs. Differently put, for every correct proof of a conclusion c from a set of premises $p_1 \ldots p_n$, there exists an instance of the reflection schema (a proof) that describes the premises, the conclusion, and the sequence of proof rules applied to arrive at the conclusion; i.e., such an instance contains the *name* of a proof. To check whether a sentence is a correct instance of the reflection schema (or denotes a correct proof), Haas describes an algorithm that basically extracts (or unquotes) the sentences $p_1 \ldots p_n$ from the name of the proof used in the instance of the reflection schema, and then tries to prove the conclusion c using these premises. If such a proof exists, the instance has been shown to be correct.

Haas's theory does not contain rules of the kind that if John's beliefs entail q then John believes q. It only deals with descriptions of whether some agent's beliefs entail a certain sentence or not (as described by instances of the reflection schema). The question of whether an agent itself or an agent it simulates will actually form a new belief is determined by running the theorem prover.

Discussion

One of the major advantages of *syntactic* treatments of belief is that they do not suffer from the problem of logical omniscience, as do semantic or modal theories of belief. This advantage, however, comes at considerable cost. It stems from the fact that belief is modeled as a relation between an agent and a sentence (a syntactic object). These sentences are denoted by fancy constants such as the quoted lists described above. The constants serve simply as names for the actual sentences, which is why *believe* remains a standard, first-order predicate that relates two individuals. Staying within first-order logic is the second major reason why syntactic logics of belief are used. For example, consider the two sentences

(believe John 'P)
(believe John ('$\vee$ 'P 'Q)).

We stayed with Haas's notation, but we could have just as well used simple strings for the belief objects. Now, suppose the first sentence is true under a particular interpretation, which models the

[1] A *wff* is a well-formed formula.

case that John believes P. This does *not* imply that the second sentence has to necessarily be true also under that interpretation, for the same reason that if "(believe John a)" is true it does not follow that "(believe John b)" has to be true also. Thus, if John believes P he does not necessarily believe $P \vee Q$, which is a logical consequence of P; hence, according to this syntactic logic of belief he is not logically omniscient, since he does not automatically believe all the logical consequences of his base beliefs. Another problem that is handled by this approach is that the rule of substitutivity of equals for equals automatically fails in belief contexts (the way it should), since nothing can be substituted inside the indivisible strings that serve as objects of belief sentences.

The cost we talked about above is that this is a very unintuitive model of belief, since it models it as a relation between agents and strings. Suppose we used such a language to represent Cassie's beliefs about other agents. This means that the language she uses at the top level to represent her "plain" beliefs would be different from the language that she uses inside belief contexts. In a sense, the strings used inside the *believe* predicate would really only be some strange gibberish to her, since they are in a completely different language from her primary language of thought; thus, she would not be able to understand such belief objects at all.

Another cost, maybe less severe, is that syntactic logics are notationally complex. Moreover, since it is usually desirable to do inference based on the structure of sentences within belief contexts, a complicated quoting and unquoting machinery in conjunction with a complicated proof procedure of the kind described above has to be used.

An advantage of staying within first-order logic is that it allows the use (at least in principle) of standard theorem-proving techniques for the modeling of inference and reasoning. However, the necessary quoting and unquoting, possibly at multiple levels, might make such techniques "even more intractable" than they already are.

Haas does not give any details about the actual structure of an agent's reasoning engine (or theorem prover). He also avoids the problem of dealing with the defeasibility of the simulation process by only dealing with "sentences entailed by some agent's beliefs." However, in the section where he describes how a robot would use the reflection schema to simulate some other agent's reasoning, it is implied that, after a certain conclusion has been proved via simulation, the robot would believe that the simulated agent believes that conclusion. In this case, an incomplete or inconsistent agent model would have similarly fatal consequences to those we described for Konolige's deduction model of belief.

3.6 Arbab's Propositional Surrogates

Arbab [1988; 1989] extends a formalism originally developed by Church in order to resolve the "paradox of the name relation" without giving up classical first-order predicate logic or the axioms of equality. The paradox arises in formalizations of identity statements in conjunction with sentences that contain words such as *know, believe, seek, search, necessary, want*, etc., which are said to create *indirect* or *opaque contexts.*

Here are some examples of paradoxical conclusions resulting from unqualified substitution of equals for equals in indirect contexts: From *it is necessary that* $\sqrt{81} = 9$ and that $\sqrt{81} =$ *the number of planets*, it paradoxically follows that *it is necessary that the number of planets = 9.* Similarly, from *John believes that the morning star is blue* and *the morning star is the evening star*, it paradoxically follows that *John believes that the evening star is blue.*

Arbab's solution is to use a *proposition surrogate* as the second argument in predicates used to formalize statements such as *John wants..., John believes...*, etc., instead of the proposition or sentence itself. A proposition surrogate is an n-tuple that contains a lambda expression to represent the form of the proposition, and the set of primitive, function, and predicate constants used in it; i.e., it is an abstraction with respect to the constants contained in the proposition. Proposition surrogates provide a proposition-encoding mechanism that is similar in spirit to quotation schemes used in syntactic theories of belief such as Haas's [Haas, 1986]. A nice feature of proposition surrogates, however, is that the back-transformation into the sentences they encode is achieved by simply applying the rules of lambda conversion.

Using Arbab's modification of Church's algorithm to compute the proposition surrogate of *the evening star is red* (formalized as *red(estar)*), we get

$$\langle \lambda P \lambda S \langle *P, *S \rangle, @red, @estar \rangle$$

which can then be used to form

$$believes(john, \langle \lambda P \lambda S \langle *P, *S \rangle, @red, @estar \rangle)$$

to represent *John believes that the evening star is red.* *@red* and *@estar* are understood to be pointers to the actual primitive constants, and * is a de-referencing operator. Now, if we know that $mstar = estar$, the paradoxical conclusion *John believes that the morning star is red* can be avoided, because $mstar = estar$ does not imply that $@mstar = @estar$.[2]

Arbab then goes on to use proposition surrogates in a formulation of the Wise Man puzzle (cf. page 154) and develops a Prolog program that can automatically solve the puzzle. However, in order to simulate the reasoning of the wise men, he has to explicitly formulate that they know how to apply modus ponens, which is done in the following way (the notation $\boxed{\text{P}}$ stands for the proposition surrogate of P; (x) stands for universal quantification of x):

$$\begin{aligned} &(x)(y) x \neq y \supset \\ &\quad ((know(x, \boxed{know(y, \boxed{W(x)})}) \wedge \\ &\quad know(x, \boxed{know(y, \boxed{W(x)}) \supset W(x)})) \supset \\ &\quad know(x, \boxed{W(x)})) \end{aligned}$$

Note that this is a very specific application of modus ponens and not the formulation of a general inference rule. In his formulation of the puzzle, Arbab has to anticipate every single reasoning step and "precompile" an appropriate inference rule such as the one shown above. For example, going one level deeper in the belief nesting would require a whole new set of rules dealing with reasoning at that level. Arbab is really not concerned with the problem of simulative reasoning at all, which might be why his implementation of a solution for the Wise Man puzzle is very specifically geared towards the puzzle and does not generalize to other simulative reasoning problems.

3.7 ViewGen

ViewGen is a program that implements the theory of belief ascription developed by Ballim and Wilks et al. [Wilks and Ballim, 1987; Ballim *et al.*, 1991; Ballim and Wilks, 1991; Ballim, 1992].

[2]Note that $\langle \lambda P \lambda S \langle *P, *S \rangle, @red, @estar \rangle \neq red(estar)$ because the terms are of different type (n-tuple vs. truth value), even though the proposition surrogate evaluates to $red(estar)$ by the rules of lambda conversion.

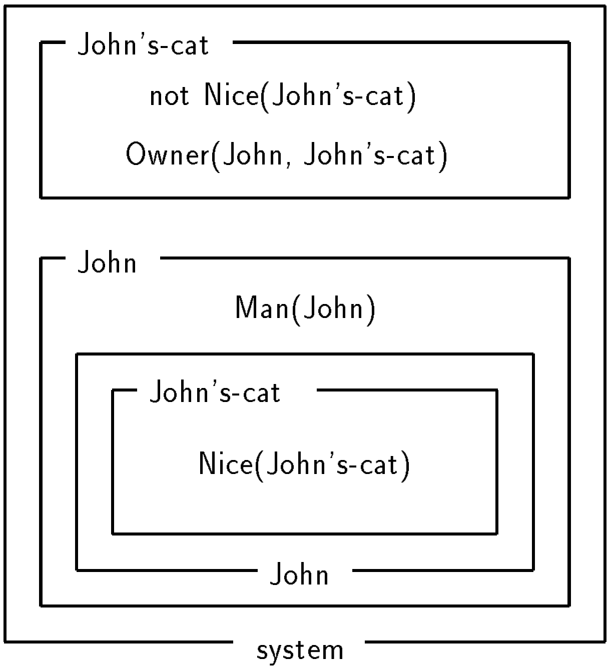


Figure 3.1: A set of *ViewGen* environments

Its main purpose is to generate topic-specific nested points of view that can be used by other programs that analyze discourse, perform planning, etc. The main data structure of *ViewGen* is an *environment*, which serves as a container or database to hold various propositions relevant to a specific *topic*, or that represent the *viewpoints* a certain agent has about various topics. Topic and viewpoint environments can be arbitrarily nested, and they are all embedded in one global environment which represents the system's point of view.

Figure 3.1 shows an example set of environments in *ViewGen*. Rectangles with labels at their top left corner are used to represent topic environments; labels at the bottom center indicate viewpoint environments. The system's beliefs about John are contained in the topic environment labeled "John"; in particular, the beliefs the system believes John holds are contained in a nested viewpoint environment also labeled "John", which resides in the topic environment describing John. *ViewGen*'s default rule for belief ascription is that everybody's view on things is the same as the system's, unless there is evidence to the contrary. Only such differing beliefs are explicitly stored in viewpoint environments, which explains why in the example in Figure 3.1 the system's belief about John's cat causes John's viewpoint environment to be non-empty.

To be able to represent *atypical* beliefs, such as secrets, expertise, self-knowledge, etc., which would be inappropriately ascribed to other agents by the default ascription rule, Ballim and Wilks introduce *lambda formulas* as a representational device for propositions. A lambda formula consists of a lambda expression and an *evaluation relation* which explicitly (or implicitly via class membership) defines the set of agents competent to evaluate the lambda expression. The evaluation relation also describes what value the lambda expression evaluates to for a certain agent. This can be used, for example, to represent differing beliefs of agents about somebody's phone number. The belief ascription algorithm is then changed such that lambda expressions are only ascribed to agents if they are explicitly or implicitly mentioned in the evaluation relation of that lambda expression.

Discussion

Ballim et al. [1991] call *ViewGen* and the computational model underlying it "a highly pragmatic approach" that focuses on a common-sense plausible reasoning scheme about propositional attitudes. They argue that managing groups of propositions in explicit environments facilitates rapid decision making in realistic discourse understanding. A drawback, however, is that the use of environments to represent beliefs of cognitive agents is a variant of what Moore [1977] calls the database approach to belief representation. Moore's analysis shows (cf. page 18) that this approach suffers from serious problems, for example, difficulties in how to represent non-belief as in "John does not believe p", or disjunctive belief as in "John either believes p or he believes q", and, hence, is a somewhat limited device for the representation of nested propositional attitudes.

ViewGen does not perform any reasoning with other agent's beliefs. Its primary focus is on a form of belief ascription, where the system's beliefs are ascribed to other agents by "pushing propositions inwards" while checking for beliefs that are different from the system's point of view. This process which is called *ascription by perturbation* is then combined with a process that works on evaluation relations of the kind described above, in order to account for proper ascription or shadowing of private beliefs, expertise, etc. Ballim [1992] goes on to develop the beginnings of a framework called *ViewFinder* for representing and reasoning with nested beliefs and other attitudes. In this framework he presents a somewhat complex "menagerie of environments" (Section 9.2, p. 134) and a logical language to talk about them. This environment language alleviates some of the representational shortcomings of *ViewGen* described above. Various problems such as belief maintenance in nested environments and various ascription operators are discussed within this framework, and some possible solutions are proposed. However, while *ViewGen* is actually implemented, *ViewFinder* does not have an implementation. It is only seen as an open-ended framework that can serve as a starting point for further investigation.

3.8 ATT-Meta

ATT-Meta [Barnden, 1992; Barnden *et al.*, 1994a; Barnden *et al.*, 1994b] is a system that combines *simulative* and *metaphor-based* reasoning in order to aid in the analysis of certain discourse-comprehension phenomena. It is claimed that certain metaphorical commonsense models of the mind such as "mind as container" or "ideas as internal utterances" are used frequently in spoken and written discourse in order to describe mental states and processes of participating agents, and that such metaphorical descriptions are particularly useful for understanding the discourse. ATT-Meta is a system that can reason about mental states reported in small fragments of discourse, with a particular emphasis on the analysis of any metaphorical descriptions involved.

ATT-Meta uses a first-order episodic logic with equality as its basic representation device. The logic is *episodic*, since terms can denote "episodes", which are understood to be any sort of situation, event, or process. For example, the episode that John kissed Mary during some time interval τ would be represented as

`#ep(Kissing,` τ`, John, Mary).`

In this representation, `Kissing` is taken to denote the set of all possible kissing episodes, and the remaining arguments specify during what time interval τ the kissing took place and what other entities were involved. Objects in the domain such as the kissing episode can be *non-certain* or *certain*, where non-certain objects are interpreted by ATT-Meta as being not necessarily real. It is not entirely clear what is meant by "real" here, but a possible conjecture is that objects that are certain do have an actual or real denotation in a particular domain of discourse, while non-certain objects do not necessarily have such a denotation. For example, the sentence

`#certain(#ep(Kissing,` τ`, John, Mary))`

would represent to ATT-Meta that the particular kissing event actually took place. For non-certain objects and episodes, three degrees of uncertainty can be specified by means of the predicates `#possible`, `#suggested`, or `#presumed`, where `#possible` is taken to have the lowest degree of certainty.

In ATT-Meta, belief is represented in the form of nested episodes. For example, a belief such as the one reported in "Bill believes that John was ill on `<some date>`" would be represented (assuming it is certain) as

`#certain(#ep(#Believing-Certain, now, Bill, #ep(Being-Ill, John,` τ`)))`

where τ denotes the time interval for the specified date, `now` denotes some time interval including the current instant, and `#Believing-Certain` denotes the set of all conceivable episodes of an agent believing something with certainty. Thus, this belief is represented as an episode of Bill being in a particular relationship with the episode of John being ill on some date. Different degrees of Bill's belief can be expressed independently from ATT-Meta's degree of certainty with special kinds of episodes such as `#Believing-Certain`, `#Believing-Presumed`, etc.

What is considered to be ATT-Meta's most important aspect of representation is the way metaphorically described belief states are expressed. However, since we are mainly interested in the way ATT-Meta represents and uses non-metaphorical belief in simulative reasoning, we will not discuss the representation and handling of such metaphors.

The reasoning of ATT-Meta is directed by a set of production rules of the form

`<LHS>` $\longrightarrow$ `[<qualifier>] <RHS>`,

where `<LHS>` is a list of episode-denoting terms, `<RHS>` is one such term, and `<qualifier>` is one of `suggested`, `presumed`, or `certain`. These production rules link episodes to episodes, and they are pursued in a backward-chaining, goal-directed fashion. For example, a rule describing that every boy who loves somebody during some time τ is, presumably, happy during that time would be stated as

`#ep(Loving,` τ`,` x`,` y`), #ep(Being-Boy,` τ`,` x`)` $\longrightarrow$ `[presumed] #ep(Happy,` τ`,` x`).`

If, then, during the course of reasoning, ATT-Meta becomes interested in whether a particular individual was happy at some time, it would match the right-hand side of this rule and generate two subgoals for the episodes listed on the left-hand side. The qualifier `[presumed]` specifies the maximum certainty for the derived result. In case any of the facts supporting the left-hand side have lesser certainties than that, the lowest such certainty would be attributed to the derived result.

To carry out belief reasoning with episodes that represent the beliefs of other agents, ATT-Meta uses *simulative reasoning*. Whenever it becomes interested in whether a particular agent a believes

p, it sets up what is called a *simulation pretense cocoon* (similar to SIMBA's simulation contexts) and tries to infer p as well as its negation within that cocoon by simulating a's reasoning. For every subgoal q created during that simulation inference, the following subgoals are set up outside the cocoon:

1. `#ep(#Believing-`ρ`, `τ`, `a`, `q`)`
2. `#ep-neg(#ep(#Believing-`ρ`, `τ`, `a`, `q`))`
3. `#ep(#Believing-`ρ`, `τ`, `a`, #ep-neg(`q`))`
4. `#ep-neg(#ep(#Believing-`ρ`, `τ`, `a`, #ep-neg(`q`)))`

`#ep-neg` is a function that yields the negation of an episode, and ρ stands for the various certainty predicates that generate different degrees of belief, such as `#Believing-certain`, etc. If any of these outside subgoals succeeds, its object episode (or proposition) is imported into the simulation cocoon to enable it to participate in the simulation of the agent. Once the original goal p has been derived, it is exported from the cocoon and attributed to the simulated agent as one of its beliefs. This exportation is done *defeasibly*, by qualifying the resulting episode with one of the non-certainty predicates described above. The rules for computing the certainty of the derived result are somewhat complex; hence, we will not repeat them here. Within a simulation, all of ATT-Meta's production rules that govern its own reasoning are available, but only those episodes (or propositions) will be used for which a corresponding belief episode of the simulated agent is available. In addition to trying to derive p via simulation, ATT-Meta also tries to derive the corresponding belief episodes `#ep(#Believing-`ρ`, `τ`, `a`, `q`)` directly, outside the simulation cocoon, with any information available there.

Discussion

ATT-Meta is a system which is very close in spirit to SIMBA. The development of the two systems has gone on independently and in parallel for some time, which might explain the pursuit of similar goals and ideas. ATT-Meta meets many of the desiderata described in Section 2.11. The main differences between SIMBA and ATT-Meta are the underlying belief model, the way the basic reasoning rules are defined for the system, and the way in which the defeasibility of simulation results is handled.

One characteristic of the representation scheme used by ATT-Meta which we find problematic if used in an artificial cognitive agent such as Cassie is that essentially two different languages are used to describe the propositions believed by ATT-Meta, on the one hand, and the propositions believed by agents ATT-Meta knows about, on the other hand. As mentioned above, ATT-Meta's representations are expressions in a first-order logic extended with special terms to denote episodes. Following an example in [Barnden *et al.*, 1994a], a possible representation for the episode that John kissed Mary lovingly at some time τ would be this:

`e = #ep(Kissing, `τ`, John, Mary)` $\wedge$ `manner(e) = lovingly`

That is, the extra information about the manner of the kissing can be specified outside of the episode as an additional restriction using the representational apparatus of the underlying first-order logic, and we assume that this is the way one would represent such an episode at the top level of ATT-Meta.[3] However, to represent that Bill believes (now) that John kissed Mary lovingly (at some time τ), the above representation cannot be used to represent the object of Bill's belief, since belief is represented as a relation between an agent and an *episode*, while the above is a *sentence* in the underlying first-order logic. The only way to represent such special kissings as objects of belief is to encode their manner as a new type of `Loving-Kissing` episode, for example,

`#certain(#ep(#Believing-Certain, now, Bill, #ep(Loving-Kissing,` τ`, John, Mary))),`

similar to the way `#Believing-Certain` had to be used to represent the certainty of the believing episode, instead of using the `#certain` predicate used at ATT-Meta's top level. Assuming this representation, ATT-Meta's own reasoning about kissings needs to be different from its simulative reasoning about kissings or loving kissings that other agents have beliefs about, which seems to counteract the elegance and efficiency of simulative reasoning. The only other solution we see is to restrict any top-level assertions and predications about episodes to the various certainty predicates, thus keeping the representational vocabulary used at the top level and within belief contexts more or less identical. In that case, however, the representational power of the underlying first-order logic would not be utilized very much at all, and representational distinctions would mainly be made by using different kinds of episode terms.

In a similar vein, special episode functions such as `#ep-conj`, `#ep-disj`, `#ep-neg`, etc., which combine episodes into compound episodes, have to be used to represent within belief contexts what the logical connectives $\wedge, \vee, \neg$, etc., represent at the top level. The connection between these episode connective functions and ordinary logical connectives remains unclear. A related question is how belief about universal propositions such as "Fred believes that all dogs are happy" would be represented, even though it is claimed that special representations for quantificational episodes such as "all dogs being happy over some interval τ" are available.

Another important difference between SIMBA and ATT-Meta is the set of available reasoning rules and the way they are specified. SIMBA uses a somewhat minimalistic, logic-based approach, by defining Cassie's reasoning in terms of a small set of inference rules of a deductive system for the logic **SL**. These rules (at least for a monotonic variant of the logic) are provably sound relative to the chosen semantics. In ATT-Meta, reasoning is defined with a set of production rules that link episodes to episodes. An exact and complete specification of these rules is not provided in the publications available to us, but we assume that it is a much bigger set than the set of inference rules used by SIMBA, even though probably not all rules of SIMBA have equivalents in ATT-Meta.

For example, a reasoning rule used in an example in [Barnden *et al.*, 1994a] is paraphrased as "if a body of instructions is wrong it is not good to follow it". Using a richer set of reasoning rules is not a problem in itself, since simulative reasoning is not dependent on any particular such set. One possible problem we see, however, is that rules such as the above, which are not completely domain-independent, implicitly encode domain knowledge that would better be represented explicitly. A more extreme example would be a rule that captured some form of mathematical-reasoning expertise. This is specifically important, since ATT-Meta's reasoning rules are attributed identically to all agents during simulation (similar to what is done by SIMBA); hence, such domain expertise might be attributed to agents who are not experts in the particular domain.[4] With the

[3]The reason this is an assumption is that the publications available to us are not very explicit about the representations used at the top level of ATT-Meta; instead, their focus is on the representation of particular episodes and their use in the representation of, and reasoning about, belief.

[4]A method for ascribing ATT-Meta's reasoning rules more discriminately is currently being developed (Barnden, personal communication).

minimal set of rules used by SIMBA, however, such accidental attribution cannot take place, while special expertise can still be formulated as a set of domain propositions which can then be used selectively during the simulation of experts. Another advantage of a smaller, well-defined set of rules is that it makes the system easier to analyze.

Some other aspects we consider important that are missing from ATT-Meta are the ability to reason with hypothetical belief, e.g., as in "if Bill believed p, would he believe q?", the ability to distinguish belief from belief entailment, e.g., as in "Bill's beliefs entail q even though he does not believe it", and a formal specification of a semantics for the representation language. On the other hand, there are various features of ATT-Meta that are not available in SIMBA, for example, reasoning about time, and metaphorical belief reasoning.

3.9 Nested Theorist

Nested Theorist [van Arragon, 1991] is a user-modeling tool aimed at modeling a user's reasoning with incomplete information. Nested Theorist (NT) is based on Theorist [Poole *et al.*, 1987], which is a simple framework for default reasoning. NT extends Theorist to allow default reasoning at arbitrarily many levels. The main objective of NT is to provide a formal tool that can reason with incomplete information in the form of defaults and that can reason about users reasoning with such defaults.

Theorist uses as its input two sets of formulae: a set $\mathcal{F}$ called *facts*, and a set Δ called *defaults*. For example, here is how Theorist can be used as a simple user modeler (cf. [van Arragon, 1991, p. 266]):

$$\mathcal{F} = \{\ novice(david)\ \}$$
$$\Delta = \{\ understands_A(login) \leftarrow novice(A)\ \}$$

In this example, Theorist has a fact that David is a computer novice and that novices usually understand the login command. If in Theorist some formula g is a logical consequence of its facts and consistent defaults, then it is said that Theorist can *explain* g. Explanations are defined as sets $\mathcal{F} \cup D$, where D is some set of ground instances of Δ. In the above example, the set

$$\mathcal{F} \cup \{\ understands_{david}(login) \leftarrow novice(david)\ \}$$

is an explanation of $understands_{david}(login)$, thus, Theorist can explain that David understands the login command. If, then, an observation of David would add another fact describing that he could not login, and Theorist would have the additional fact that agents who cannot login do not understand the login command, then the above set would not be an explanation for $understands_{david}(login)$ any more, since it would be inconsistent with one of the facts.

Nested Theorist distinguishes between two levels of facts and defaults: The level of the *system s* described by $\mathcal{F}$ and Δ, and the level of the *user u* described by $\mathcal{F}_u$ and Δ_u. The facts and defaults of the system are represented as before, but the facts and defaults of the user are represented with the help of a special *meta-language* that is used to represent object-level facts and defaults of the user, and with predicates $\mathcal{F}_u$ and Δ_u that assert membership of user sentences in the associated sets of the same name. For example, s could have the following set of facts:

$$\mathcal{F} = \{\ \mathcal{F}_u(\ forall(x', and(p'(x'), q'))\ )\ \}$$

As a notational convenience, this is usually written as

$$\mathcal{F} = \{ \ \mathcal{F}_u(\ \forall X \ \ p(X) \wedge q \ ) \ \},$$

since everything within $\mathcal{F}_u$ and Δ_u is assumed to be a meta-language expression. In the implementation, however, a Prolog meta-interpreter needs to be used to reason with these meta-language sentences that describe facts and defaults of the user.

Using a strategy very similar to its own to reason with the sets of facts and defaults it believes are believed by the user, NT can then determine whether the user believes certain consequences by trying to find explanations for them from the sets $\mathcal{F}_u$ and Δ_u. The only additional complication is that NT cannot really know whether a particular set of ground instances of Δ_u is consistent with the facts believed by the user, since it does not assume that $\mathcal{F}_u$ constitutes complete knowledge of the user's beliefs. To work around that, NT uses a Consistent Assumptions Default, which means that if it cannot determine whether a particular set of assumptions is inconsistent with $\mathcal{F}_u$, it will assume by default that they are in fact consistent.

An extension of NT called Prioritized Nested Theorist uses a system of *prioritized defaults* to deal with multiple explanations that are caused by conflicting defaults. Another extension called Limited Nested Theorist can model some forms of limited reasoning based on a method of *linear resolution*. For example, it can model users that can only perform inferences that take less then a certain number of steps.

Discussion

NT is not intended to be a particular theory of knowledge and belief, but many of the design goals cited for NT are also design goals for SIMBA, for example, that reasoning can be nested to arbitrary depth, that there may be many agents at each level of nesting, that an agent's representation of other agents is incomplete, that logical inference is used as a model for reasoning, and that the underlying representation scheme needs to be general.

One of our main criticisms is with this last point: Even though NT is a logic-based approach, the way beliefs of other agents such as the user u are treated is akin to the database approach to belief representation (cf. page 18), thus it is not as general as one would desire. The reason is that reasoning with the beliefs of u is bound to the set $\mathcal{F}_u$; however, such a set is not uniquely defined any more in the case of belief disjunction, e.g., if it is known that u either believes p or q. In that case NT would need to define two new "users", u_p and u_q, that would represent the alternative sets of beliefs. Similar problems arise with negative belief, e.g., if it is known that u does not believe p. The description of NT does not mention the possibility of quantifying into belief contexts, but doing so would also be complicated, since NT uses a syntactic, object-language/meta-language approach. Other aspects not handled by NT are reasoning with hypothetical beliefs, and the distinction between belief and belief entailment.

The theory of NT is not restricted to just one level of belief nesting as described above. Thus, according to the theory, many agents could reason about each other's beliefs at various levels of nesting. The implementation as described in [van Arragon, 1991], however, only handles one level of nesting. There is no restriction that would prevent an extension to multiple levels, but the object-language/meta-language approach would probably make that prohibitively expensive, since every step at a particular object level requires several steps at the meta-level, which in turn would require several steps per step at the meta-meta-level, etc.

The emphasis of NT is on reasoning about other agent's reasoning with defaults, which is something not done by SIMBA. The way SIMBA uses default reasoning is at the belief ascription stage; that is, the result of a simulation is ascribed to another agent by way of default. However, the derivation of such a result will not involve any default reasoning in the standard sense.

3.10 Summary

To describe what could be called the state of the art in belief ascription, let us briefly summarize the various approaches discussed above:

One group of them concentrates on representational issues. Various devices get developed to handle standard problems in the representation of iterated propositional attitudes. Creary and Maida suggest how their representation formalisms enable reasoning about the mental states of others by way of simulation; however, no computational account for this kind of reasoning is given. Arbab goes a small step further by developing a special-case Prolog solution to a puzzle that uses simulative reasoning, but his solution cannot be easily generalized.

Moore, Konolige, and Haas concentrate on extending first-order predicate logic to capture the notions of knowledge or belief. Proof procedures for these logics model the reasoning about the mental states of others. Problems with the formalization of non-monotonicity lead them to either neglect the issues of defeasibility inherent in this kind of reasoning (e.g., Konolige) or to avoid them all together (e.g., Moore and Haas).

Viewgen, ATT-Meta, and Nested Theorist are some of the very few systems around that are actually capable of ascribing beliefs to other agents. *Viewgen* does so very simplistically, more or less ignoring issues of defeasibility and belief revision. ATT-Meta is probably the closest match with the set of desiderata we set forth. Our main criticisms of it are the use of a somewhat unintuitive belief model and the lack of a full formal specification and semantics. Issues such as hypothetical belief reasoning and representation of belief entailment are also not handled. A similar set of criticisms applies to Nested Theorist.

Summing up, there is only a small number of implemented systems around that can actually do some form of simulative reasoning. None of them, however, meets all the objectives we deem necessary to serve as the foundation of an artificial cognitive agent such as Cassie.

Chapter 4

A Fully Intensional, Subjective Logic of Belief

In this chapter we will give a precise definition of the basic ingredients of our belief logic **SL**. As motivated in Chapter 2, **SL** is intended to be the fundamental building block of the belief system of an artificial cognitive agent such as Cassie. Since its language $L_{\mathbf{SL}}$ is the representational substrate for things in Cassie's mind, it has to be a *belief representation language* as opposed to a *world representation language* (see [Maida and Shapiro, 1982, p. 296] for a discussion of this distinction). Due to this point of view, the design of $L_{\mathbf{SL}}$ is guided by the following principle assumptions which are derived from [Maida and Shapiro, 1982] and [Shapiro, 1993]:

- The domain of discourse, $\mathcal{D}$, whose elements are denoted by terms of $L_{\mathbf{SL}}$, is a set of *intensional entities* such as propositions, objects of thought, fictional objects, etc.
- $\mathcal{D}$ consists of a set of atomic objects, $\mathcal{D}_a$, without internal structure, and a set of structured objects, $\mathcal{D}_s$, whose structure derives in some well-defined way from more primitive objects.
- Two distinct terms of $L_{\mathbf{SL}}$ cannot denote the same entity, a restriction which is called the *uniqueness principle*, because every object is denoted by a unique term.

Since we take intensions or sets of intensional objects as a completely independent realm, as opposed to, for example, the notion used by Montague where intensions are defined in terms of sets of extensions [Montague, 1974], we call our approach *fully intensional* (see [Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1991] for more discussion on this view of intensionality). One reason for this approach is that it provides elegant solutions to problems of indirect reference, such as McCarthy's telephone number problem [McCarthy, 1979; Maida and Shapiro, 1982], or problems of representation with *de re* and *de dicto* belief reports [Rapaport, 1986].

We call **SL** a *subjective* logic, since $L_{\mathbf{SL}}$ is intended to be Cassie's language of thought in which she represents and reasons, as opposed to being a language to describe the beliefs of other agents from an objective, outside observer's point of view.

Finally, the uniqueness principle is not so much a requirement as a reflection of how we view cognitive function: Whenever Cassie creates a new (mental) term (of $L_{\mathbf{SL}}$) for some object of $\mathcal{D}$, she does so because no already existing term denotes that object for her; hence, it must be an intensionally different object. Had it been the same, the preexisting term would have been used as a consequence of cognitive economy.

4.1 $\mathbf{SL}_0$: A Monotonic Precursor to SL

SL is a nonmonotonic logic that supports belief revision. For matters of exposition, we will develop a simplified monotonic version called $\mathbf{SL_0}$ first, which will present the fundamental concepts and tools necessary to formalize **SL**. The main difference between $\mathbf{SL_0}$ and **SL** is in the definition of the deductive system, and in some of the semantics associated with belief. The full logic **SL** will be described in Chapter 5.

The core idea in the formalization of **SL** is derived from [Shapiro, 1993]: It is that $L_{\mathbf{SL}}$ is primarily a language of terms, not of sentences. The vehicles for constructing these terms are proposition-valued functions. For example, an expression such as $\mathsf{B(Mary, B(Sally, Cute(John)))}$ is simply a nested function application which yields a proposition as its result, and not a higher-order expression as it would be in a standard treatment where B is viewed as a relation. Logical connectives, the main sentence constructors in standard logical treatments, are just a special subset of these functions. As motivated in Chapter 2, sentences of **SL** are only used to single out those propositional terms in Cassie's mind, which represent propositions believed by her. Sentences simply represent a belief flag. If a particular propositional term p is part of Cassie's mind, and if its belief flag is set, we will say her mind contains the sentence $!p$. In $\mathbf{SL_0}$ this will be synonymous with saying that Cassie believes the proposition denoted by p.

Notational conventions: We will use sans serif for object-language terms, *italics* for meta-variables ranging over such terms, **bold sans serif** for domain objects, and ***bold italics*** for meta-variables ranging over domain objects. The denotation relation is expressed with the usual double brackets, for example, $[\![\mathsf{Man(Hans)}]\!]=\mathbf{Man(Hans)}$, $[\![p_1]\!]=\boldsymbol{p_1}$. $f(x)\downarrow$ will mean that f is defined for x, $f(x)\uparrow$ that it is not.

4.2 $L_{\mathbf{SL}_0}$: The Language of $\mathbf{SL}_0$

$L_{\mathbf{SL}_0}$ is an internal language; hence, it is or at least can be different for every agent. However, there is a certain structure that we require for every instance of $L_{\mathbf{SL}_0}$. For that reason, we describe the various sets of symbols defined below as sets of metavariables. Where appropriate, typical object-language instances of these variables are given.

Definition 4.2.1. *The atoms of $L_{\mathbf{SL}_0}$ are defined as the following sets of symbols:*

1. $P =_{def} \{(,),,,\forall,\exists,!\}$*, punctuation, quantifiers, assertion*
2. $I =_{def} \{i_1, i_2, i_3, \ldots\}$*, the set of individual constants, e.g.,* $\mathsf{Mary}, \mathsf{B}_1, \mathsf{B}_2, \ldots$
3. i_e*, the ego or self concept, e.g.,* I
4. $I_p \subset I$*, the set of individual propositional constants*
5. $F^n =_{def} \{f_1^n, f_2^n, f_3^n, \ldots\}$*, the set of n-ary function constants*
6. $F =_{def} \bigcup_{n\geq 1} F^n$*, the set of all function constants, e.g.,* $\mathsf{Loves}, \mathsf{Knows}_1, \ldots$
7. $F_p \subset F$*, the set of propositional function constants*
8. $F_l =_{def} \{f_\neg, f_\wedge, f_\vee, f_\Rightarrow\}$*, the set of logical connective functions,* $F_l \subset F_p$*, e.g.,* $\neg$, $\wedge$, $\vee$, $\Rightarrow$
9. f_B*, the belief function, e.g.,* B
10. $V =_{def} \{x_1, x_2, x_3, \ldots\}$*, the set of variables, e.g.,* $\mathsf{x}, \mathsf{y}_2, \mathsf{dog}, \ldots$

Now that we have defined all the atomic elements of $L_{\mathbf{SL}_0}$ we can start to define complex elements such as terms and sentences. In order to be able to specify quantified terms, we have to define substitutions first:

Definition 4.2.2. *Let e be an expression of $L_{\mathbf{SL}_0}$. The* **substitution** $e_{x/i}$ *is the expression obtained from e by replacing all occurrences of x in e that are not in the scope of a quantifier (free occurrences) with i.*

Definition 4.2.3. *The set of terms, T, is defined by the following inductive rules:*

1. *Every $i \in I$ is a term. If $i \in I_p$, then i is a propositional term.*
2. *If $t_1, \ldots, t_n$ are terms and $f^n \in F$, then $f^n(t_1, \ldots, t_n)$ is a function term. If $f^n \in F_p$, then $f^n(t_1, \ldots, t_n)$ is a propositional function term.*
3. *If x is a variable and t is a propositional function term with an individual i as a subterm such that no occurrence of i is in the scope of a quantifier $\forall x$ or $\exists x$, then $\forall x\, t_{i/x}$ and $\exists x\, t_{i/x}$ are propositional terms.*

Definition 4.2.4. *Let T_p be the set of propositional terms. Then $S =_{def} \{!t | t \in T_p\}$ is the set of all sentences.*

As mentioned before, the only role of sentences is to flag those propositional terms in the mind of the agent under consideration, which represent propositions that are actually believed by that agent. The scope of the '!' operator is always the complete proposition term following it, thus, there will always be exactly one exclamation mark per sentence. Functions are intended to generate structured names that denote structured objects, for example, the function symbol Loves is intended to denote the function which has all propositions of the form that *one individual loves another individual* as its range. The term $\mathsf{Loves}(\mathsf{John}, \mathsf{Mary})$ is intended to denote the proposition that *John loves Mary*. The term $\mathsf{B}(\mathsf{Lucy}, \mathsf{Loves}(\mathsf{John}, \mathsf{Mary}))$ is intended to denote the proposition that *Lucy believes that John loves Mary*, a proposition which contains the proposition from the previous example as a part.

For now, we assume that the structure defined by the standard syntax of function application and composition is sufficient to describe the structure of objects such as propositions. The main drawback of this scheme is that it always encodes the order of the arguments even if the resulting proposition should be order independent. The logical-connective functions provide a good example: In our current scheme the terms $\wedge(\mathsf{P}, \mathsf{Q})$ and $\wedge(\mathsf{Q}, \mathsf{P})$[1] denote two different propositions (because of the uniqueness principle), but one could make an argument that this should not be so.

4.3 $S_{\mathbf{SL}_0}$: A Semantic Account for $\mathbf{SL}_0$ via Agent Interpretations

Intuitively, Cassie's belief system is filled with a set of $L_{\mathbf{SL}_0}$ expressions; hence, $L_{\mathbf{SL}_0}$ can be viewed as Cassie's language of thought. It is an *internal* language, much like the language used inside the scope of Konolige's modal belief operator $[S_i]$ [Konolige, 1986], but $\mathbf{SL}_0$ does not define an *external* language that describes Cassie's belief system from an outside observer's point of view. Since $L_{\mathbf{SL}_0}$ is Cassie's language of thought, she uses that very language to represent other agents' beliefs, and there is no need for quotation, standard names, naming maps, etc.

As builders of Cassie, we are of course interested in how her internal language is linked to the outside. That these internal expressions denote the proper individuals and propositions is a

[1] In order to conform with standard logical notation, $\neg$ will usually be written in prefix notation, and the binary connectives will usually be written in infix notation, for example, $\neg\mathsf{Man}(\mathsf{Hans})$ instead of $\neg(\mathsf{Man}(\mathsf{Hans}))$, and $\mathsf{Man}(\mathsf{Hans}) \wedge \mathsf{Man}(\mathsf{Franz})$ instead of $\wedge(\mathsf{Man}(\mathsf{Hans}), \mathsf{Man}(\mathsf{Franz}))$. The definition of F_l given above lists the connectives in order of decreasing precedence. To be able to write unambiguous expressions without excess parentheses the usual operator precedence rules apply. Subexpressions connected by connectives of equal precedence associate to the left.

prerequisite for her being able to understand and be understood by other agents. The connection between Cassie's internal language and the external, though not extensional, domain $\mathcal{D}$ will be made via *agent interpretations.*

Definition 4.3.1. *An* **intensional domain of discourse***, $\mathcal{D}$, is a non-empty set of intensional individuals which consists of two disjoint subsets: $\mathcal{D}_a$, a set of atomic elements, and $\mathcal{D}_s$, a set of structured elements. Each of these parts is further split into a propositional part, $\mathcal{D}_{a_p}$ and $\mathcal{D}_{s_p}$ which taken together form $\mathcal{D}_p$, and a non-propositional part, $\mathcal{D}_{a_{np}}$ and $\mathcal{D}_{s_{np}}$ which taken together form $\mathcal{D}_{np}$.*

An intensional domain is intended to contain objects of thought, discourse entities, concepts, propositions, impossible objects, fictional objects, etc., all of which are denoted by terms of $L_{\mathbf{SL}_0}$. For the purposes of our exposition, it suffices to mainly distinguish between propositions and other objects. To provide proper denotations for function terms, we need actual functions that operate on $\mathcal{D}$. Because of the uniqueness principle, not just any set of functions will do. The following definition specifies some necessary characteristics for a set of domain functions that can be used in an interpretation:

Definition 4.3.2. *Let $\mathcal{D}$ be an intensional domain of discourse. A set of functions $\mathcal{F}$ is a* **basis** *for $\mathcal{D}_s$ iff*

1. *every $\boldsymbol{f} \in \mathcal{F}$ is an n-ary, injective (or 1-to-1) function $\mathcal{D}^n \rightarrow \mathcal{D}_s, n \geq 1$, and*
2. *for every $\boldsymbol{f_1}, \boldsymbol{f_2} \in \mathcal{F}$ such that $\boldsymbol{f_1} \neq \boldsymbol{f_2}$ the range of $\boldsymbol{f_1}$ is disjoint from the range of $\boldsymbol{f_2}$, and*
3. *for every $\boldsymbol{i} \in \mathcal{D}_s$ there is an n-ary function $\boldsymbol{f^n} \in \mathcal{F}$ and a set of arguments $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_i} \in \mathcal{D}$ such that $\boldsymbol{f^n}(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) = \boldsymbol{i}$.*

The conditions that the functions need to be 1-to-1 and that their ranges are required to be mutually disjoint are direct consequences of the uniqueness principle. If a particular function were not 1-to-1, then there would be two different sets of arguments for which the function would have the same value. If the ranges were not disjoint, then there would be two different functions with a value in common. If such a set of functions were to be used in an interpretation, then there would be at least two different terms of $L_{\mathbf{SL}_0}$ with the same denotation, a violation of the uniqueness principle.

Finding an actual basis of domain functions is a very hard and mainly unsolved problem. It amounts to the construction of a theory about how natural language gets mapped into propositional meaning representations. Every domain function can be viewed as a kind of case frame, and the problem then becomes to find a correct and complete set of such case frames. This problem is not unique to our approach; every standard logical approach faces a similar problem as soon as an actual domain theory has to be constructed.

The denotations of the logical connective functions and the belief function are the only ones defined by $\mathbf{SL}_0$ itself, as opposed to leaving that up to a particular interpretation:

Definition 4.3.3. *The set of* **logical connective functions***, $\mathcal{F}_l$, is the set $\{\mathsf{f}_\neg, \mathsf{f}_\wedge, \mathsf{f}_\vee, \mathsf{f}_\Rightarrow\}$ whose individual elements are defined as follows (let $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{D}_p$ be arbitrary propositions):*

$\mathsf{f}_\neg$: $\mathcal{D}_p \rightarrow \mathcal{D}_{s_p}$. *The value of $\mathsf{f}_\neg(\boldsymbol{p})$ is the proposition that it is not the case that $\boldsymbol{p}$.*
$\mathsf{f}_\wedge$: $\mathcal{D}_p^2 \rightarrow \mathcal{D}_{s_p}$. *The value of $\mathsf{f}_\wedge(\boldsymbol{p}, \boldsymbol{q})$ is the proposition that it is the case that $\boldsymbol{p}$ and $\boldsymbol{q}$.*
$\mathsf{f}_\vee$: $\mathcal{D}_p^2 \rightarrow \mathcal{D}_{s_p}$. *The value of $\mathsf{f}_\vee(\boldsymbol{p}, \boldsymbol{q})$ is the proposition that it is the case that either $\boldsymbol{p}$ or $\boldsymbol{q}$ or both.*

$\mathbf{f}_{\Rightarrow}$: $\mathcal{D}_p^2 \to \mathcal{D}_{s_p}$. *The value of* $\mathbf{f}_{\Rightarrow}(\boldsymbol{p}, \boldsymbol{q})$ *is the proposition that if it is the case that* $\boldsymbol{p}$ *then it is the case that* $\boldsymbol{q}$.

Definition 4.3.4. *The* **belief function**, $\mathbf{f}_{\mathsf{B}}$, *is defined as follows (let* $\boldsymbol{a} \in \mathcal{D}_{a_{np}}, \boldsymbol{p} \in \mathcal{D}_p$*):*

$\mathbf{f}_{\mathsf{B}}$: $\mathcal{D}_{a_{np}} \times \mathcal{D}_p \to \mathcal{D}_{s_p}$. *The value of* $\mathbf{f}_{\mathsf{B}}(\boldsymbol{a}, \boldsymbol{p})$ *is the proposition that* $\boldsymbol{a}$ *believes* $\boldsymbol{p}$.

Since we do not have a formal theory of propositions (our semantic domain), we used English sentences to define the classes of propositions yielded by the proposition-valued functions defined above. So, in a sense, English is our external language that we use to interpret Cassie's internal language. This is not all that much different from standard truth recursion rules which usually define the truth of compound expressions with the help of natural-language conjunctions whose semantic function is taken to be understood.

Terms of $L_{\mathbf{SL}_0}$ will be mapped onto the domain of discourse by means of an interpretation function which has to fulfill the following requirements:

Definition 4.3.5. *Let* $\mathcal{D}$ *be an intensional domain of discourse and* $\mathcal{F}$ *a basis for* $\mathcal{D}_s$. *A function* $int : T \to \mathcal{D}$ *is an* **admissible** $L_{\mathbf{SL}_0}$ **interpretation function** *for* $\mathcal{D}$ *and* $\mathcal{F}$ *if it satisfies the following conditions:*

1. *If* $i \in I$ *then* $int(i) \in \mathcal{D}_a$.
2. *If* $i \in I_p$ *then* $int(i) \in \mathcal{D}_{a_p}$.
3. *If* $f \in F$ *then* $int(f) \in \mathcal{F}$.
4. *If* $f \in F_p$ *then* $int(f) \in \mathcal{F}_p$.
5. $int(\neg) = \mathbf{f}_{\neg}$, $int(\wedge) = \mathbf{f}_{\wedge}$, $int(\vee) = \mathbf{f}_{\vee}$, $int(\Rightarrow) = \mathbf{f}_{\Rightarrow}$, $int(\mathsf{B}) = \mathbf{f}_{\mathsf{B}}$.
6. *If* t *is an n-ary function term of the form* $f^n(x_1, \ldots x_n)$, *then* $int(t) = int(f^n)(int(x_1), \ldots, int(x_n))$.
7. *If* t *is a term of the form* $\forall x\, t'$, *then* $int(t)$ *is the proposition that, for every atomic domain element* $int(i)$ *such that* $int(t'_{x/i})\!\downarrow$ *it is the case that* $int(t'_{x/i})$.
8. *If* t *is a term of the form* $\exists x\, t'$, *then* $int(t)$ *is the proposition that, there is an atomic domain element* $int(i)$ *such that* $int(t'_{x/i})\!\downarrow$ *and it is the case that* $int(t'_{x/i})$.
9. *For any two terms* t_1, t_2, *if* $t_1 \neq t_2$, *then* $int(t_1) \neq int(t_2)$.

Note, that domain functions are not assumed to be total, therefore they can be undefined for certain sets of arguments. For example, an interpretation function *int* might map the function constant Loves onto the domain function **Loves**, which, when applied to proper domain arguments, e.g., $\textbf{Loves}(int(\mathsf{John}), int(\mathsf{Mary}))$, would yield a proposition such as *John loves Mary.* However, in an application such as $\textbf{Loves}(int(\mathsf{Dog}(\mathsf{Fido}), int(\mathsf{Red}(\mathsf{Apple}_1)))$ its value should be undefined. This is the reason why in cases 7 and 8 above we have to make sure to only consider substitution instances whose interpretation is defined, since arbitrary substitution in a term might have taken it out of the domain of a particular domain function.

Note also, that our account of universal quantification only allows us to express generalizations over countable domains, since the quantifiable universe is construed as being contained in the range of the interpretation function *int* whose domain is countable. Thus, we could not express generalizations over uncountable domains such as the real numbers.

Now we can finally define how $L_{\mathbf{SL}_0}$ terms in the "mind" of an agent are to be interpreted:

Definition 4.3.6. *Let $a = \langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, int \rangle$ be an* **agent interpretation structure**, *where $\mathcal{D}$ is an intensional domain of discourse, $\mathcal{F}$ is a basis for $\mathcal{D}_s$, $\mathcal{B} \subset \mathcal{D}_p$ is the base set of the agent's beliefs, $\boldsymbol{e} \in \mathcal{D}_{a_{np}}$ is the agent's ego or self-concept, and int is an admissible $L_{\mathbf{SL}_0}$ interpretation function for $\mathcal{D}$ and $\mathcal{F}$. Then for every $t \in T : [\![t]\!]_a =_{def} int(t)$.*

The agent's base set of beliefs $\mathcal{B}$ is intended to describe the set of propositions believed by the agent without being justified by the logic $\mathbf{SL}_0$ itself, i.e., they serve as a set of extra-logical belief axioms. Intuitively, such beliefs are formed from sensory information, from being told by somebody, etc. Since we are concerned with the modeling of realistic agents, we will usually think of $\mathcal{B}$ as being finite.

Apart from being able to interpret what actual propositions are believed by an agent, it would be interesting to know whether the agent behaves rationally. One aspect of rational behavior is to draw proper conclusions from one's current set of beliefs. Below we present a notion of *justification* which defines whether an agent is justified in believing a certain proposition relative to some agent interpretation. We will use the notation $\models_a !p$ (a turnstile with a J-bar for 'justification') to express that an agent is justified in believing $[\![p]\!]_a$ relative to some interpretation a. Intuitively, if Cassie tells us that she believes some proposition $[\![p]\!]$, then we can verify whether she is justified in believing it by checking whether $\models !p$ holds. If Cassie tells us that she does not believe $[\![p]\!]$, then checking whether $\models !p$ is not of much help, because we do not assume that Cassie's beliefs are deductively closed. In that sense, our notion of justification characterizes the reasoning of an ideal agent, or, to use Levesque's terminology, it specifies the set of *implicit* beliefs of Cassie [Levesque, 1984].

Definition 4.3.7. *Let t, t_1, t_2 be propositional terms of $L_{\mathbf{SL}_0}$, $a = \langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, int \rangle$ be an agent interpretation structure, and i_e be the $i \in I$ such that $[\![i]\!]_a = \boldsymbol{e}$. Then we define whether an agent is justified in believing $[\![t]\!]_a$ relative to an agent model, written $\models_a !t$:*

1. *If $[\![t]\!]_a\downarrow$ and $[\![t]\!]_a \in \mathcal{B}$ then $\models_a !t$.*
2. *$\models_a !\neg t$ iff $\not\models_a !t$.*
3. *$\models_a !t_1 \wedge t_2$ iff $\models_a !t_1$ and $\models_a !t_2$.*
4. *$\models_a !t_1 \vee t_2$ iff $\models_a !t_1$ or $\models_a !t_2$.*
5. *$\models_a !t_1 \Rightarrow t_2$ iff $\not\models_a !t_1$ or $\models_a !t_2$.*
6. *$\models_a !\forall x\, t$ iff $\models_a !t_{x/i}$ for all $i \in I$ such that $[\![t_{x/i}]\!]_a\downarrow$.*
7. *$\models_a !\exists x\, t$ iff $\models_a !t_{x/i}$ for some $i \in I$ such that $[\![t_{x/i}]\!]_a\downarrow$.*
8. *$\models_a !\mathsf{B}(i_e, t)$ iff $\models_a !t$.*
9. *$\models_a !\mathsf{B}(b, t), b \neq i_e$ if there are terms $p_1, \ldots, p_n, n \geq 0$ such that $\models_a !\mathsf{B}(b, p_1), \ldots, \models_a !\mathsf{B}(b, p_n)$ and $\models_{a'} !t$ with $a' = \langle \mathcal{D}, \mathcal{F}, \{[\![p_1]\!]_a, \ldots, [\![p_n]\!]_a\}, [\![b]\!]_a, int \rangle$.*

The cases for the logical connectives in the definition above are very similar to the truth recursion rules used in standard semantics of first-order predicate logic. The main difference is the terminology, because we are not concerned with the notion of truth of sentences. This neglect for a truth-valued semantics might sound unusual to some, but it is a natural consequence of our belief model. To say that a sentence $!p$ is true can only mean that for a particular implementation of Cassie the expression p actually is part of her mind and its associated belief flag is set. Thus it would characterize Cassie's internal state from an outside observer's point of view. However, whether a certain sentence $!p$ actually is present in Cassie's mind or not is completely dependent on the underlying machinery that adds sentences to it, as well as the state of the world this machinery operates in. Thus, apart from very simple cases an accurate definition of the truth of a sentence is completely beyond our and probably any logic. The best we can do for sentences is to define the notion of justification given above. The interesting cases of our definition are discussed below:

Case 1 deals with the fact that our interpretation function simply maps propositional terms onto propositions. It does not say anything about the agent's belief status regarding these propositions. By comparison, the corresponding case in a standard first-order semantics is the one that handles the truth of ground sentences such as $\mathsf{Red}(\mathsf{Apple}_1)$. There the interpretation function maps predicate or relation names onto sets, and the truth of atomic sentences is determined by membership of the interpreted arguments in these sets.

Case 8 defines the semantics of introspection. It exemplifies best the subjective character of $\mathbf{SL}_0$, since the self belief is based on a "plain" proposition that is not itself nested inside a belief function (though it could be). This case has some similarity with a subjective version of the inference rule S4 of modal systems, but of course, S4 is an inference rule of the *deductive system* of certain modal logics, while the above is a definition of the *semantics* of introspection in $\mathbf{SL}_0$. We do not need an extra case $\models_a !\neg\mathsf{B}(i_e, t)$ iff $\not\models_a !t$ to characterize negative introspection, because that follows as a simple theorem from the definition.

Case 9 defines the semantics of simulative reasoning (cf. Section 2.7), in which an agent such as Cassie hypothetically assumes beliefs it thinks are held by other agents, and then tries to infer consequences from these beliefs with its own reasoning mechanism. The semantics of this is captured by basing the justification of a belief sentence on a variant of the agent model which only contains the beliefs of the simulated agent as the base set, and which uses the simulated agent as its ego.

There are always infinitely many interpretations under which a particular sentence is justified; hence, we normally work with the following stronger definition of justification:

Definition 4.3.8. $!p_1, \ldots, !p_n \models !q$ *with* $n \geq 0$ *iff for all agent interpretations* a *such that* $\models_a !p_1, \ldots, \models_a !p_n$ *it is the case that* $\models_a !q$.

Since the set of base beliefs $\mathcal{B}$ might contain arbitrary propositions, it is also useful to only consider *consistent* agent interpretations:

Definition 4.3.9. *An interpretation* $a = \langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, int \rangle$ *is* **consistent** *if there is no* p *such that* $\models_a !p \wedge \neg p$ *or* $\models_a !\mathsf{B}(b_1, \mathsf{B}(b_2, \ldots, \mathsf{B}(b_n, p \wedge \neg p) \ldots)), n \geq 1$.

An alternative to the consistency requirement would be to restrict $\mathcal{B}$ to primitive propositions which do not contain any logical connective functions or quantifiers; however, this would not allow us to model agents who believe in universally quantified propositions that they were simply told about.

4.4 $D_{\mathbf{SL}_0}$: A Deductive System for $\mathbf{SL}_0$

Stewart Shapiro [1991, p. 3] defines a "*full* logic to be a language, together with a deductive system and a semantics". Since deductive systems operate purely syntactically, i.e., the derivation of new sentences from antecedents is determined completely by syntactic rules, some people consider the deductive system to be a part of the syntax (or language) of a logic. We will, however, consider it in its own right.

So far we have defined two of the above-mentioned ingredients of a logic, or, to speak in terms of the mind-as-a-container metaphor, we have defined the *structure* of some of the objects that make up Cassie's mind, i.e., the terms and sentences of $L_{\mathbf{SL}_0}$, and we have defined what these objects *mean*, i.e., the intensional entities they denote. What is still missing is a specification of how Cassie's mental objects can be *manipulated* or reasoned with. This task will be accomplished by $D_{\mathbf{SL}_0}$, a deductive system for $\mathbf{SL}_0$.

Put differently, in SIMBA, reasoning is modeled as a form of logical inference or deduction. The deductive system of the logic underlying SIMBA will specify what sort of inferences Cassie will be able to make, and the implementation of a decision procedure for this deductive system will serve as her actual reasoning engine. Depending on the type of the underlying logic, such a decision procedure might not exist (e.g., even standard first-order predicate calculus is only semidecidable), in which case we have to settle for some bounded search procedure which always terminates (in some adequate amount of time), but which might not always generate a result. In the context of modeling the reasoning of a cognitive agent, this seems to be a realistic approach.

The terms *deductive* or *deduction* are used to refer to a certain style of inference from a small number of premises to a conclusion, but not to the narrower interpretation of truth-preserving inference where the truth of the conclusion follows necessarily from its premises. Konolige [1986, p. 21] characterizes deductive inference rules as rules that have the properties of *provinciality*, i.e., the number of premises of a particular rule is fixed and finite, and *effectiveness*, which means that every rule is an effectively computable function of its premises. By virtue of choosing a fairly standard logic as the main building block of SIMBA, most of the reasoning modeled by it will be deductive (in the wider sense) with the nonmonotonic belief reasoning fragment as the main exception.

Of course, even under the wider interpretation, not all reasoning is deductive; hence, a realistic version of Cassie will need additional reasoning mechanisms that are of a more heuristic nature. In SIMBA, however, we will not be concerned with heuristic or inductive forms of reasoning, mainly because they are much less understood. The exception is the simulative belief reasoning mechanism, which in itself can be viewed as a particular reasoning heuristic. Other interesting aspects of agenthood, such as how inference is tied in with action or how new symbols can be introduced by grounding them in perception, will be neglected as well. For approaches to these problems see, for example, [Kumar, 1994; Lammens, 1994].

4.4.1 An Argument for Natural Deduction

When choosing a deductive or proof system for a particular logic, one has to consider a variety of questions, for example:

- How well does it mesh with the semantics? Can we obtain soundness and completeness results?
- How complex is its specification?
- How good is it to work with for a human; i.e., how difficult is it to perform actual deductions?
- How good is it to work with for a computer; i.e., how amenable is it to automation?
- How good is it to work with for doing metatheory; i.e., how does it influence the complexity of proofs about the logic itself?
- How closely does it match the structure of valid logical arguments carried out by humans?

- How much is already known about a particular system; how well is it understood?

The actual design goals derived from the questions above very much depend on the task at hand. For example, do we want to do automatic theorem proving, do we want to prove certain properties of a formal system, do we want to formalize logical, mathematical, or philosophical arguments, or do we want to model the reasoning of a cognitive agent? No particular deductive system performs well at all of the above; hence, one needs to choose one that performs strongly in what is most important for the problem one wants to solve.

Since we want to use a deductive system to model the reasoning of a cognitive agent, it is desirable that the individual inference steps of that system make some cognitive sense. That they be cognitively *valid* would be ideal, but, since this is too ambitious a goal, we will settle for the somewhat vague requirement of "closeness" to human reasoning. Thus, our choice is to use a *natural deduction* system, e.g., [Fitch, 1952] or a variation thereof, since such systems attempt to formalize the structure of logical arguments as they are "naturally" carried out by humans. In the automatic-theorem-proving community, natural deduction systems are often classified under the category *human-oriented* systems, e.g., [Bledsoe, 1977; Loveland, 1984; Plaisted, 1990], a characterization which also supports our choice of such a system to model the reasoning of a cognitive agent.

Natural deduction systems usually have introduction and elimination rules for each logical connective, as well as some special-purpose rules such as the rule of hypothesis. This abundance of rules makes the specification somewhat complex and the metatheory lengthy, because many of the proofs have as many cases as there are inference rules. These disadvantages of natural deduction systems are compensated by their higher adequacy to model the reasoning of cognitive agents. Possible alternatives would be systems based on *resolution* [Robinson, 1965] or *analytic tableaux* [Smullyan, 1968], but while such systems are well suited for automatic theorem proving, they are certainly farther away from being a cognitively valid model of human reasoning than a system based on natural deduction.

4.4.2 An Argument for Reasoning Contexts

Our deductive system is strongly influenced by the system of Shapiro and Wand (or SW) [Shapiro and Wand, 1976], the system SWM (for Shapiro, Wand, and Martins) [Martins, 1983; Martins and Shapiro, 1988], and the system SWMC (for Shapiro, Wand, Martins and Cravo) [Cravo and Martins, 1993a]; hence, we will quickly review some of their important characteristics. As indicated by the names, the three systems constitute a logical progression, but their differences go beyond mere extensions and are quite substantial. Collectively, we will call them the SW* systems.

The system SW is a logic system based on one of the relevance logics developed by Anderson and Belnap called FR [Anderson and Belnap, 1975]. The motivation for the development of SW was to have a logic system useful for natural language understanding and question answering applications. The main innovation of SW was to completely eliminate the representation of subproof structure as it was generated by derivations in the natural deduction system FR. Instead, every well-formed formula (wff) has an *origin set* associated with it which represents the set of hypotheses on which its derivation was based. It also has an *origin tag* which records whether the wff is a hypothesis or derived. Together the origin tag and the origin set represent the support of the wff. With this representation of support the (relative) truth of a wff becomes independent of a particular derivation. As long as all the hypotheses in its support are held true, the wff is held true also. In SWM Martins developed this concept further and exploited it in the implementation of a belief revision system and the Multiple Belief Reasoner (MBR). Cravo later added the representation and reasoning with defaults to arrive at SWMC.

What is common to all SW* systems, is the notion of a supported wff (swff). Such an swff is usually written as a triple $\langle p, \tau, \omega \rangle$, where p is the actual wff, τ is the origin tag which indicates whether the wff is a hypothesis or derived, and ω is the origin set containing the hypotheses on which the derivation of p was based.

As described above, one of the main design goals behind SW was to eliminate the representation of subproof structure. Thus, the SW* systems completely separate the specification of individual inference rules from the contexts in which these inference rules may be usefully applied. For example, the rule of implication-elimination (or modus ponens) is specified like this:

From $\langle p, \tau_1, \omega_1 \rangle$ and $\langle p \Rightarrow q, \tau_2, \omega_2 \rangle$ we may infer $\langle q, \mathtt{der}, \omega_1 \cup \omega_2 \rangle$.

This specification is completely context independent. It does not say anything about the fact that it only makes sense to apply this rule if both premises are true, or assumed to be true, or asserted, or believed. Instead, a *contextual interpretation* is defined completely independently which determines whether a certain proposition is considered in the application of an inference rule or not. For example, in the definition of the Multiple Belief Reasoner, Martins [1983, p. 4.4] defines a *context* as a set of hypotheses and a *belief space* determined by such a context as the set of hypotheses defining the context plus all the propositions derived exclusively from them. At any point, the set of all hypotheses currently believed is called the *current context*, and a proposition is said to be *believed* if it belongs to the belief space determined by the current context or, equivalent to that, if its origin set is a subset of the current context.[2] For the application of an inference rule, then, only those propositions are considered which are believed in the current context.

Since the SW* systems do not explicitly represent contexts (or subproof structure), their deductions are completely flat and unstructured. The set of hypotheses that define the current context has to be specified externally (or by the implementation); it is not represented at the logic level in an individual deduction or proof. This is not a big problem as long as only a small number of contexts have to be considered. However, if one wants to model *simulative* reasoning, which is a form of *hypothetical* reasoning, this lack of context structure quickly becomes a problem, because many different hypothetical and simulation contexts might have to be considered in an individual reasoning sequence. For this reason, we will introduce *deduction* or *reasoning contexts* as the main structuring device of the deductive system of $\mathbf{SL}_0$. Additionally, we will associate every sentence with a *support*, just as in the various SW* systems.

4.4.3 An Introductory Example

Informally, an $\mathbf{SL}_0$ deduction is a system of linked deduction contexts. Every deduction context describes a reasoning subspace which contains a possibly empty set of hypotheses and a set of sentences that were derived from these hypotheses with the use of deduction rules such as and-elimination, modus ponens, etc., which will be defined below. All but the top-level deduction context contain a link to their parent context, from which they can *import* sentences, and to which they *export* results. During import and export, sentences undergo transformations which depend on the type of the deduction context. The traditional terms for import and export are *reiteration* and *hypothesis discharge*, but these terms are mainly used for the process of implication introduction,

[2]This definition of *believed proposition* is circular unless we assume that hypotheses are *primitively* believed.

while we want to use a more general terminology that can also be applied to simulative deductions. The two types of deduction contexts are *simulation contexts*, which serve to simulate the reasoning of a particular agent, and *hypothetical contexts*, which serve to do standard hypothetical reasoning necessary for implication introduction. Both types of contexts are similar in that they model a kind of hypothetical reasoning. They differ, however, in the way they import from parent contexts and in the way they export results. The top-level deduction context is a simulation context which models Cassie's very own reasoning.

	Cassie ($\top$), I	
1	!Dog(Fido)	Hypothesis
2	!B(Mary, $\forall$x (Dog(x) $\Rightarrow$ Animal(x)))	Hypothesis
		open context Hyp
10	!B(Mary, Dog(Fido)) $\Rightarrow$ B(Mary, Animal(Fido))	$\Rightarrow$Introduction, 3, 9

	Hyp (Cassie), I	
3	!B(Mary, Dog(Fido))	Hypothesis
4	!B(Mary, $\forall$x (Dog(x) $\Rightarrow$ Animal(x)))	Reiteration, 2
		open context Cassie/Mary
9	!B(Mary, Animal(Fido))	Belief introduction, 8

	Cassie/Mary (Hyp), Mary	
5	!Dog(Fido)	Simulation hypothesis, 3
6	!$\forall$x (Dog(x) $\Rightarrow$ Animal(x)))	Simulation hypothesis, 4
7	!Dog(Fido) $\Rightarrow$ Animal(Fido)	Universal instantiation, 6
8	!Animal(Fido)	$\Rightarrow$Elimination, 5, 7

Figure 4.1: A simple $\mathbf{SL}_0$ deduction

Figure 4.1 shows a simple example which demonstrates the general structure of $\mathbf{SL}_0$ deductions. Deduction contexts are drawn as boxes that contain a sequence of supported $\mathbf{SL}_0$-sentences. To keep things simple, the supports are omitted in this example. Simulation contexts have double vertical lines; hypothetical contexts just have single lines. Every context has a unique identifier, a pointer to its parent context in parentheses, and the actual agent of that context written on top of it. The top-level context does not have a parent context, which is indicated with the $\top$ (or "top") symbol in its parent slot. Every $\mathbf{SL}_0$ sentence is associated with a unique step number written on the left and with an explanation that justifies it (the individual deduction rules will be spelled out in detail below). To follow the deduction, one has to follow the step numbers in sequence. This is slightly unusual for a natural deduction system — usually, subdeductions are displayed with nesting or as branches of a tree — but giving subdeductions "static extent" makes it possible to reenter them, pursue them in parallel, and do other interesting things necessary once we do non-monotonic reasoning.

Let us quickly step through the deduction shown in Figure 4.1: The top of the top-level contex box contains the name of the context "Cassie", its pointer to the parent context in parentheses is $\top$, and I, Cassie's self-concept, is the reasoning agent of the context. Step 1 represents Cassie's belief that Fido is a dog, and step 2 represents her belief that Mary believes that all dogs are animals. These beliefs are not justified by any other beliefs; hence, they have the status of belief *hypotheses*. What we want to show is that, from these hypotheses, Cassie can infer the conditional that if Mary believes that Fido is a dog, then she also believes that he is an animal. To perform the necessary hypothetical reasoning, Cassie opens the hypothetical context Hyp, where in step 3 she assumes (for the sake of argument) that Mary believes that Fido is a dog. Step 4 is nothing but a technicality which imports the hypothesis of step 2 into the context Hyp. Now Cassie has to simulate Mary's reasoning. To do that, she opens a simulation context Cassie/Mary, in which she assumes the object propositions of the two belief sentences in the parent context Hyp as simulation hypotheses. What follows is a simple reasoning sequence from Fido is a dog (step 5) and all dogs are animals (step 6) to Fido is an animal (step 8). This result of step 8 can now be attributed to Mary in step 9. Now we can discharge the hypothesis of the context Hyp by introducing the entailment in step 10 of the top-level context which gives us the result we were after.

4.4.4 General Characteristics of $D_{\mathbf{SL}_0}$

$D_{\mathbf{SL}_0}$ is defined as a notational natural deduction system. By *notational* is meant that its rules are primarily about the writing down of certain objects and sentences in a structured way on a piece of paper. This is not really a restriction; it just makes it easier to do the metatheory. The structures used by $D_{\mathbf{SL}_0}$ suggest a straightforward mapping onto data structures usable by a computer program. How that is done is described in more detail in Chapter 6.

As already mentioned, the central structuring objects of $D_{\mathbf{SL}_0}$ are *deduction* or *reasoning contexts*, which are drawn as boxes as shown in Figure 4.1. A particular $D_{\mathbf{SL}_0}$ deduction is a drawing that contains a linked collection of deduction contexts.

A deduction context is drawn as a box with a headline and a body. The headline specifies the following information:

1. The *name* of the context, which has to be unique for a particular deduction.
2. The name of the *parent context*, or the $\top$ symbol if it does not have a parent.
3. The *agent* whose reasoning is carried out, which has to be some $L_{\mathbf{SL}_0}$ individual $i \in I$.

The body contains a finite sequence of supported $L_{\mathbf{SL}_0}$ sentences, each associated with a deduction step number on the left and an explanation on the right.

There are two types of deduction contexts:

1. *Simulation* contexts, in which the basic reasoning of a particular agent is carried out.
2. *Hypothetical* contexts, in which hypothetical reasoning of a particular agent is carried out.

Simulation and hypothetical contexts differ in the way they import and export information from and to their parent contexts. Simulation contexts are drawn with two vertical lines to distinguish them from hypothetical contexts. Figure 4.2 shows a deduction context with name 'Name', parent context 'Parent', agent 'Agent', and a simple sequence of body sentences. On the left, the

	Name (Parent), Agent
1	$!\mathsf{P}$
2	$!\mathsf{P} \Rightarrow \mathsf{Q}$

	Name (Parent), Agent
1	$!\mathsf{P}$
2	$!\mathsf{P} \Rightarrow \mathsf{Q}$

Figure 4.2: Simulation and hypothetical contexts

context is drawn as a simulation context, on the right it is drawn as a hypothetical context. Note, that in many of the examples to follow where we are only interested in a particular deduction or reasoning sequence, we will simply use propositional constants such as P, Q, etc., instead of "real" propositions. This is of course not a restriction of the logic, it is only done to save space.

Carrying out a $D_{\mathbf{SL}_0}$ deduction is a repeated application of deduction steps. Every deduction step belongs to one of the following two principle classes:

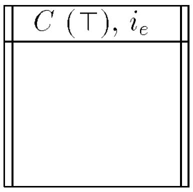
1. Opening a new deduction context.
2. Adding an $L_{\mathbf{SL}_0}$ sentence to a particular deduction context according to some deduction rule.

The rules specified below govern when and how these steps are to be carried out. Since $D_{\mathbf{SL}_0}$ is a notational system, every rule specification has a graphical illustration of the object to be drawn in a particular step and an explanation of the various constraints that are not immediately obvious from the illustration.

4.4.5 Context Rules

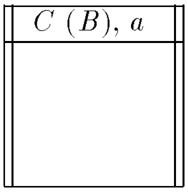
The following rules specify when and how deduction contexts can be opened. By *opening* is meant adding an appropriate box to the current deduction figure.

Opening the Top-Level Context



A $D_{\mathbf{SL}_0}$ deduction is started by opening an empty simulation context to carry out Cassie's reasoning. This first context of every deduction is called the *top-level* context. It is given some arbitrary name C, an empty parent context $\top$, and the agent i_e (Cassie's ego or self concept).

Opening a Simulation Context



Once the top-level context has been opened, new, empty simulation contexts can be opened. Every new context is given an arbitrary name C that is unique in the current deduction, the name of an already existing context B as its parent context, and any $L_{\mathbf{SL}_0}$ individual $a \in I$ as its agent which must be unique among all simulation contexts that have B as their parent context. This uniqueness condition ensures that there is no more than one simulation context per agent at any level of nesting.

Opening a Hypothetical Context

$C\ (B), a$

Once the top-level context has been opened, new, empty hypothetical contexts can be opened. Every new context is given an arbitrary name C that is unique in the current deduction, the name of an already existing context B as its parent context, and the agent of its parent context as its agent.

It should be evident that the rules above generate a completely connected system of deduction contexts. In every particular deduction, all but the top-level context have a link to their parent context. Similarly, for every context, we can find all its children by simply looking for all contexts that have it as their parent.

4.4.6 Deduction Rules

Deduction rules govern how new sentences can be added to deduction contexts. Said differently, they govern how new sentences can be *deduced* from previous sentences. Typically, a deduction rule specifies a possibly empty set of premises, more exactly, patterns of premises, and a conclusion that can be inferred if the specified premises hold at the current point of the deduction. Deduction rules operate on *supported* sentences, which are defined as follows:

Definition 4.4.1. *A* **supported sentence** *is a quadruple $\langle !p, a, \tau, \omega \rangle$, where (1) $!p \in S$ is an $L_{\mathbf{SL}_0}$ sentence, (2) a is either some agent $\in I$ or the unspecified agent ?, (3) τ is an origin tag, which can be either* `hyp` *or* `der`*, and (4) $\omega \subseteq S$ is the origin set, which is the set of sentences on which the derivation of $!p$ is based.*

This notion of sentence support is derived from the SW* systems described above. The tuple notation of a supported sentence can be viewed as an in-line notation for a single step of an $\mathbf{SL}_0$ deduction. It describes in a succinct way how a particular sentence came about, by stating whether a particular agent's reasoning was used, whether it is a hypothesis or a derived sentence, and on what other sentences the derivation of this sentence was based. The full power of sentence supports will only be utilized in the nonmonotonic version of the logic. However, it is easier to introduce this concept now and build on it later.

Within a deduction context, we will not use the tuple notation, but instead simply add the support elements of a particular sentence as a list to the right. For example, a possible contextual notation of the supported sentence $\langle !\mathsf{Q}, \mathsf{Mary}, \mathtt{der}, \{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}\rangle$ is shown below on the left, while a possible notation of $\langle !\mathsf{Q}, ?, \mathtt{der}, \{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}\rangle$ is shown on the right:

	$C\ (\ldots)$, Mary
	...
m	$!\mathsf{Q}$, I, `der`, $\{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}$

	$C\ (\ldots)$, Sally
	...
m	$!\mathsf{Q}$, `der`, $\{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}$

The first supported sentence explicitly specifies Mary as its reasoning agent. This means that somewhere in its derivation an introspective rule must have been used. As we will see later, introspective rules encode a particular reasoning agent into their results. For that reason, the first sentence can only appear in contexts with reasoning agent Mary. Within a context, an I-flag is written right after the sentence to indicate that for the derivation of the sentence on that line an introspective rule has been used somewhere on the way. The second sentence specifies an arbitrary reasoning agent, hence, it could appear in any deduction context, for example, one whose reasoning agent is Sally. For such sentences the I-flag is omitted.

The conventions used in the definition of the various deduction rules presented below are best explained with help of an example definition:

Example: And Introduction ($\wedge$I)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p,$	$\tau_1,$	ω_1	
	$\ldots$			
n	$!q,$	$\tau_2,$	ω_2	
	$\ldots$			
$l+1$	$!p \wedge q,$	`der`,	$\omega_1 \cup \omega_2$	$\wedge$I m, n
$l+1$	$!q \wedge p,$	`der`,	$\omega_1 \cup \omega_2$	$\wedge$I m, n

This rule says that, from any two sentences $!p$ and $!q$, we are allowed to deduce their conjunction. The possible conclusions are identified by the step number $l+1$, where l is the largest step number in the deduction right before the application of this rule, or 0 if this is the first step. Since "$\wedge$" is commutative, there are two possible conclusions. Rather than displaying them in two separate contexts, we simply list them in sequence. All sentences other than the conclusions are the necessary premises which must have step numbers less than $l+1$. The order of the premises is irrelevant; in this example, they might even be identical (even though they are listed separately), since $!p$ and $!q$ are sentence patterns which can be instantiated by any well-formed sentence. The origin tag of the conclusions is `der`, their origin set is the union of the origin sets of the premises, and the explanations on the right show the abbreviation symbol for and-introduction and the step numbers of the premises involved. And-introduction can be performed in any deduction context; thus, the context is shown as half hypothetical/half simulation context. The parent context is also irrelevant, which is indicated with the "$\ldots$" symbol. The I-flag is not displayed unless handled specially by the deduction rule under consideration. The general rule is that if any of the premises has the I-flag set, then it will also be set for the conclusion.

4.4.6.1 Standard Rules

The deduction rules of this section are called *standard* rules, because they are more or less variations of the rules given in standard natural deduction treatments of first-order predicate calculus. The main difference to such standard treatments is the specification and use of sentence supports and the use of deduction contexts instead of nested subproofs or subproof trees.

Hypothesis (H)

	$C\ (\ldots), a$			
	$\ldots$			
$l+1$	$!p,$	`hyp`,	$\{!p\}$	H

At any point of a deduction, we are allowed to add some new hypothesis p to a deduction context. Since a hypothesis is not justified by any other sentence, it gets an origin set containing just the hypothesis itself. Said differently, a hypothesis only depends on itself, which is the reason for calling it a hypothesis in the first place. Hypotheses serve two primary functions: (1) to model Cassie's base beliefs which she acquired by way of perception or some other extra-logical way, and (2) as actual hypotheses used for hypothetical reasoning.

The sets of hypotheses and sentences visible at a certain point of a deduction context are defined as follows:

Definition 4.4.2. *Let C be an arbitrary deduction context with parent context D. Then we define $hyps(C,m)$ as the set of sentences $!p$ which were introduced into C via the rules of hypothesis or simulation hypothesis* (H *or* SH) *at step numbers $\leq m$. Taking into account the parent contexts of C, we define*

$$hyps^*(C,m) = \begin{cases} hyps(C,m) & \text{if } D = \top \\ hyps(C,m) & \text{if } C \text{ is a simulation context} \\ hyps(C,m) \cup hyps^*(D,m) & \text{otherwise} \end{cases}$$

Similarly, we define $shyps(C,m)$ and $shyps^*(C,m)$ as the sets of simulation hypotheses (SH) *and $sent(C,m)$, and $sent^*(C,m)$ as the sets of arbitrary sentences visible in context C at step number m.*

And-Introduction ($\wedge$I)

$$\begin{array}{r|llll|l} & C\ (\ldots),\ a & & & & \\ \hline & \ldots & & & & \\ m & !p, & \tau_1, & \omega_1 & & \\ & \ldots & & & & \\ n & !q, & \tau_2, & \omega_2 & & \\ & \ldots & & & & \\ l+1 & !p \wedge q, & \texttt{der}, & \omega_1 \cup \omega_2 & & \wedge\mathrm{I}\quad m,n \\ l+1 & !q \wedge p, & \texttt{der}, & \omega_1 \cup \omega_2 & & \wedge\mathrm{I}\quad m,n \end{array}$$

For completeness, we repeat this rule discussed in the introductory example: From any two sentences p and q we are allowed to deduce their conjunction.

And-Elimination ($\wedge$E)

$$\begin{array}{r|lll|l} & C\ (\ldots),\ a & & & \\ \hline & \ldots & & & \\ m & !p \wedge q, & \tau, & \omega & \\ & \ldots & & & \\ l+1 & !p, & \texttt{der}, & \omega & \wedge\mathrm{E}\quad m \\ l+1 & !q, & \texttt{der}, & \omega & \wedge\mathrm{E}\quad m \end{array}$$

Or-Introduction ($\vee$I)

$$\begin{array}{r|lll|l} & C\ (\ldots),\ a & & & \\ \hline & \ldots & & & \\ m & !p, & \tau, & \omega & \\ & \ldots & & & \\ l+1 & !p \vee q, & \texttt{der}, & \omega & \vee\mathrm{I}\quad m \\ l+1 & !q \vee p, & \texttt{der}, & \omega & \vee\mathrm{I}\quad m \end{array}$$

Or-Elimination ($\vee$E)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p \vee q,$	$\tau_1,$	ω_1	
	$\ldots$			
n	$!p \Rightarrow r,$	$\tau_2,$	ω_2	
	$\ldots$			
o	$!q \Rightarrow r,$	$\tau_3,$	ω_3	
	$\ldots$			
$l+1$	$!r,$	$\texttt{der},$	$\omega_1 \cup \omega_2 \cup \omega_3$	$\vee$E m, n, o

Note again that the order of the premises is irrelevant; hence, any order would have justified the application of this rule.

Double-Negation-Introduction ($\neg\neg$I)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p,$	$\tau,$	ω	
	$\ldots$			
$l+1$	$!\neg\neg p,$	$\texttt{der},$	ω	$\neg\neg$I m

Double-Negation-Elimination ($\neg\neg$E)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!\neg\neg p,$	$\tau,$	ω	
	$\ldots$			
$l+1$	$!p,$	$\texttt{der},$	ω	$\neg\neg$E m

Negation-Introduction ($\neg$I)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p \wedge \neg p,$	$\tau,$	$\omega \cup \{!h\}$	
	$\ldots$			
$l+1$	$!\neg h,$	$\texttt{der},$	$\omega \setminus \{!h\}$	$\neg$I m

From a contradiction, we can deduce the negation of any $!h \in \omega$, i.e., the negation of any hypothesis on which the derivation of the contradiction was based. Note that requiring $!h$ to be an element of the origin set of step m is not really a restriction, because we can always infuse an arbitrary hypothesis into the origin set of the contradiction by performing the following sequence of steps:

$$
\begin{array}{r|lll||l}
\hline
 & C\ (\ldots),\ a & & & \\
\hline
 & \ldots & & & \\
m & !p \wedge \neg p, & \tau, & \omega & \\
m+1 & !h, & \texttt{hyp}, & \{!h\} & \mathrm{H} \\
m+2 & !p \wedge \neg p \wedge h, & \texttt{der}, & \omega \cup \{!h\} & \wedge\mathrm{I}\quad m, m+1 \\
m+3 & !p \wedge \neg p, & \texttt{der}, & \omega \cup \{!h\} & \wedge\mathrm{E}\quad m+2 \\
 & \ldots & & & \\
l+1 & !\neg h, & \texttt{der}, & \omega & \neg\mathrm{I}\quad m+3 \\
\hline
\end{array}
$$

Therefore, in effect we do have an equivalent to the standard contradiction-elimination rule

$$p \wedge \neg p \Rightarrow q$$

at our disposal. However, what we presented as a feature, i.e., that we could infuse an arbitrary hypothesis into the origin set of a derived result, is considered a serious misfeature by relevance logicians. They go to great lengths to avoid the introduction of such irrelevant hypotheses, either by restricting the rule of $\wedge$I to conjuncts with identical origin sets, as done, for example, in the system FR summarized in [Anderson *et al.*, 1992, p. xxvi] and the system of [Shapiro and Wand, 1976], or by marking the results of $\wedge$I with a special origin tag, as done by [Martins, 1983].

All deduction rules presented so far are *simple* or *linear*, in the sense that they all only involve a single deduction context. The rule of reiteration is the first *complex* or *cross-contextual* one, because it involves two contexts.

Reiteration (R)

$$
\begin{array}{r|lll||}
\hline
 & C\ (\ldots),\ a & & \\
\hline
 & \ldots & & \\
m & !p, & \tau, & \omega \\
 & \ldots & & \\
\hline
\end{array}
\qquad
\begin{array}{r|lll|l}
\hline
 & D\ (C),\ a & & & \\
\hline
 & \ldots & & & \\
l+1 & !p, & \tau, & \omega & \mathrm{R}\quad m \\
\hline
\end{array}
$$

If D is a hypothetical context with parent context C, then we can identically reiterate (or copy) into it any sentence from its parent context. Reiteration is a complex deduction rule which defines how to *import* sentences from a parent context into a hypothetical context. It is simply a form of abbreviation which allows one to reuse a previously derived result which is based on hypotheses currently assumed in the parent context without having to rederive it in an auxiliary subproof just for the sake of making it available in the hypothetical child context.

In fact, one could avoid the rule of reiteration completely by extending the notion of premise availability in a deduction rule. Instead of requiring that all premises must be available in the same context as the conclusion, one could relax that to only requiring that they be available in the set $sent^*(C, l)$.

Generalized Reiteration (R^*)

	$C\ (\ldots), a$
	$\ldots$
m	$!p, \quad \tau,\ \omega$
	$\ldots$

	$D\ (\ldots), b$	
	$\ldots$	
$l+1$	$!p, \quad \tau,\ \omega$	$\mathrm{R}^*\ \ m$

Since origin sets record the set of hypotheses on which the derivation of a particular sentence is based, we can generalize the rule of reiteration as follows: If $\omega \subseteq hyps^*(D, l)$, then we can identically copy $!p$ into D. If $!p$ has the I-flag set, we have to additionally require that $a = b$, i.e., that the reasoning agents of the two contexts are identical. This last condition insures that sentences that were derived with self-referential rules such as introspection do not get imported into contexts where the self-referential rules would not lead to the same results.

Using generalized reiteration, we can copy any sentence from any source to any target context as long as the origin set of the reiterated sentence is a subset of the currently assumed hypotheses of the target context. In essence, this captures the contextual interpretations given by the SW* systems described above, where a sentence is said to be believed in a context iff all hypotheses in its origin set are believed in that context.

Generalized reiteration is a derived rule; i.e., it can easily be shown that any deduction using it can be transformed into an equivalent deduction that just uses the standard form of reiteration.

To aid the following exposition, we define some useful notation:

Definition 4.4.3. *Let* $P = \{p_1, \ldots, p_n\}, n \geq 0$.

$$P \Rightarrow q \equiv_{def} \begin{cases} q & \text{if } n = 0 \\ (p_1 \wedge \ldots \wedge p_n) \Rightarrow q & \text{if } n > 0 \end{cases}$$

Implication-Introduction ($\Rightarrow$I)

	$C\ (\ldots), a$	
	$\ldots$	
$l+1$	$!(hyps(D,m) \cap \omega) \Rightarrow q,\ \ \texttt{der},\ \ \omega \setminus hyps(D,m)$	$\Rightarrow$I $\ m$

	$D\ (C), a$
	$\ldots$
m	$!q, \quad \tau,\ \omega$
	$\ldots$

Implication-introduction is the inverse operation of reiteration. It allows one to *export* sentences from a hypothetical context to its parent context. Since the derivation of q might have used some of the hypotheses in context D, we have to relativize it to those hypotheses in ω that came from $hyps(D, m)$ before we can import it into the parent context. Here are two simple examples that demonstrate how implication introduction can be applied:

	Cassie ($\top$), I			
1	!P,	hyp,	{!P}	H
8	!(Q $\wedge$ R) $\Rightarrow$ (P $\wedge$ Q $\wedge$ R),	der,	{!P}	$\Rightarrow$I 7

	Hyp (Cassie), I			
2	!Q,	hyp,	{!Q}	H
3	!R,	hyp,	{!R}	H
4	!S,	hyp,	{!S}	H
5	!P,	hyp,	{!P}	R 1
6	!Q $\wedge$ R,	der,	{!Q, !R}	$\wedge$I 2,3
7	!P $\wedge$ Q $\wedge$ R,	der,	{!P, !Q, !R}	$\wedge$I 5,6

Note that a sentence will be considered by the function *hyps* only if it was introduced into the context with the rule of hypothesis, but not necessarily if it has origin tag hyp. The next example shows that we allow the degenerate case where the set of antecedents of the introduced "implication" is empty:

	Cassie ($\top$), I			
1	!P,	hyp,	{!P}	H
3	!P,	der,	{!P}	$\Rightarrow$I 2

	Hyp (Cassie), I			
2	!P,	hyp,	{!P}	R 1

Implication-Elimination or Modus Ponens ($\Rightarrow$E)

	C $(\ldots)$, a			
	$\cdots$			
m	$!p$,	τ_1,	ω_1	
	$\cdots$			
n	$!p \Rightarrow q$,	τ_2,	ω_2	
	$\cdots$			
$l+1$	$!q$,	der,	$\omega_1 \cup \omega_2$	$\Rightarrow$E m, n

Universal-Introduction ($\forall$I)

	C $(\ldots)$, a			
	$\cdots$			
$l+1$	$!\forall x\, q_{in/x}$,	der,	ω	$\forall$I m

	D (C), a		
	$\cdots$		
m	$!q$,	τ,	ω
	$\cdots$		

Conditions:

- D is a hypothetical context with parent context C.

- $hyps(D, m) = \emptyset$
- $i_n \in I$ is a new individual that does not appear in any element of $sent^*(C, l)$.
- x is free in q.

Note that requiring $hyps(D, m) = \emptyset$ is not really a restriction, because we could have raised any number of hypotheses in another hypothetical context that had D as its parent. A variation of the rule of universal introduction that does not have that requirement and, hence, combines universal introduction and implication introduction into one step is given below. However, since it is easy to show that every deduction that uses only one form of the rule can be transformed into an equivalent deduction that only uses the other form, we will stick with the simpler version given above.

Universal-Introduction′ ($\forall$I′)

	$C\ (\ldots),\ a$	
	$\ldots$	
$l+1$	$!\forall x\,((hyps(D,m) \cap \omega) \Rightarrow q)_{i_n/x}$, `der`, $\omega \setminus hyps(D,m)$	$\forall$I m

	$D\ (C),\ a$
	$\ldots$
m	$!q$, τ, ω
	$\ldots$

Conditions:

- D is a hypothetical context with parent context C.
- $i_n \in I$ is a new individual that does not appear in any element of $sent^*(C, l)$.
- x is free in $hyps(D, m) \Rightarrow q$.

Universal-Elimination ($\forall$E)

	$C\ (\ldots),\ a$	
	$\ldots$	
m	$!\forall x\, p$, τ, ω	
	$\ldots$	
$l+1$	$!p_{x/i}$, `der`, ω	$\forall$E m

Existential-Introduction ($\exists$I)

	C (...), a			
	...			
m	$!p,$	$\tau,$	ω	
	...			
$l+1$	$!\exists x\, p_{i/x},$	`der`,	ω	$\exists$I m

Condition:

- x is free in p

Existential-Elimination ($\exists$E)

	C (...), a			
	...			
m	$!\exists x\, p,$	$\tau_1,$	ω_1	
	...			
$l+1$	$!q,$	`der`,	$\omega_1 \cup \omega_2$	$\exists$E m, n, o

	D (C), a			
	...			
n	$!p_{x/i_n},$	`hyp`,	$\{!p_{x/i_n}\}$	H
	...			
o	$!q,$	$\tau_2,$	ω_2	
	...			

Conditions:

- D is a hypothetical context with parent context C.
- $hyps(D, n) \setminus \{!p_{x/i_n}\} = \emptyset$, i.e., $!p_{x/i_n}$ is the only hypothesis on which the derivation of q was based.
- $i_n \in I$ is a new individual that does not appear in q or any element of $sent^*(C, l)$.

Similar to $\forall$I$'$, we could have formulated a version of existential elimination without the requirement that p_{x/i_n} be the only hypothesis on which the derivation of $!q$ is based.

4.4.6.2 Belief Rules

This section describes the various deduction rules that deal with belief sentences. Let us begin with a set of rules that handle different forms of introspective reasoning:

Positive Introspection (+IS)

	C (...), a				
	...				
m	$!p,$		$\tau,$	ω	
	...				
$l+1$	$!\mathsf{B}(a, p),$	$\mathsf{I},$	`der`,	ω	+IS m

If some agent a believes an arbitrary proposition p, then it is justified in explicitly believing that it believes that proposition. Since this rule encodes the reasoning agent of context C into its result, the I-flag gets set in the conclusion. Note the subjectivity expressed by the form of this rule, since the premise is an ordinary sentence directly believed by the agent, and not itself a belief sentence about a, as it would be in an objective treatment that needs to describe a's beliefs from an outside point of view.

Negative Introspection ($-$IS)

$$
\begin{array}{r|llll|}
\hline
 & C\ (\ldots),\ a & & & \\
\hline
 & \ldots & & & \\
m & !\neg p, & & \tau, & \omega \\
 & \ldots & & & \\
l+1 & !\neg \mathsf{B}(a,p), & \mathsf{I}, & \texttt{der}, & \omega \\
 & & & & \\
\hline
\end{array}
\quad -\mathrm{IS}\ \ m
$$

If an agent believes a negative proposition, then it is justified in believing that it does not believe the positive version of that proposition. Since this rule encodes the reasoning agent of context C into its result, the I-flag gets set in the conclusion.

Internalization (IN)

$$
\begin{array}{r|llll|}
\hline
 & C\ (\ldots),\ a & & & \\
\hline
 & \ldots & & & \\
m & !\mathsf{B}(a,p), & & \tau, & \omega \\
 & \ldots & & & \\
l+1 & !p, & \mathsf{I}, & \texttt{der}, & \omega \\
 & & & & \\
\hline
\end{array}
\quad \mathrm{IN}\ \ m
$$

If an agent believes that it believes a certain proposition, then it can internalize that proposition and believe it directly. Since this rule is only applicable if the reasoning agent of context C is the same as the agent in the belief sentence, the I-flag gets set in the conclusion.

The last two rules of $D_{\mathbf{SL}_0}$ describe *simulative reasoning*, that is, how one agent can simulate the reasoning of another:

Simulation Hypothesis (SH)

$$
\begin{array}{r|lll|}
\hline
 & C\ (\ldots),\ a & & \\
\hline
 & \ldots & & \\
m & !\mathsf{B}(b,p_i), & \tau, & \omega_i \\
 & \ldots & & \\
 & & & \\
\hline
\end{array}
\qquad
\begin{array}{r||lll||}
\hline
 & D\ (C),\ b, & & \\
 & p_1/\omega_1,\ldots,p_i/\omega_i & & \\
\hline
 & \ldots & & \\
l+1 & !p_i, & \texttt{hyp}, & \{p_i\} \\
 & & & \\
\hline
\end{array}
\quad \mathrm{SH}\ \ m
$$

If D is a simulation context with parent context C and reasoning agent b, then we can import object propositions p_i from belief sentences $!\mathsf{B}(b,p_i)$ of the parent context into the simulation context. The imported sentences $!p_i$ are called simulation hypotheses, because they are hypothetically assumed by the simulating agent a in order to simulate b's reasoning. Every imported $!p_i$ gets a singleton origin set just like an ordinary hypothesis, and we associate it with the origin set ω_i of the source belief sentence in the top field of the simulation context. This mapping will be used by the rule of belief-introduction to construct origin sets of simulation results.

The double vertical lines used to identify simulation contexts are supposed to symbolize an importation barrier that can only be overcome by the rule of simulation hypothesis. This barrier around simulation contexts is in contrast with plain hypothetical contexts, where all sentences from all their parent contexts can be imported unchanged by the rule of reiteration.

Belief-Introduction (BI)

$$
\begin{array}{r|l|l r|l|}
\cline{2-2}
 & C\ (\ldots),\ a & & & \\
\cline{2-2}
 & \cdots & & & \\
l+1 & !\mathsf{B}(b,q),\ \ \texttt{der},\ \ \bigcup_{p_i\in\omega}\omega_i & \mathsf{BI}\ \ m & & \\
\cline{2-2} \cline{5-5}
 & & & & D\ (C),\ b, \\
 & & & & p_1/\omega_1,\ldots,p_n/\omega_n \\
\cline{5-5}
 & & & & \cdots \\
 & & & m & !q,\quad \tau,\ \ \omega \subseteq shyps(D,m) \\
 & & & & \cdots \\
\cline{5-5}
\end{array}
$$

Once we have derived a simulation result $!q$ whose origin set $\omega \subseteq shyps(D, m)$, that is, the derivation of q is only based on simulation hypotheses, we can introduce $!\mathsf{B}(b, q)$ in the parent context, or, said differently, we can *ascribe* the belief of q to agent b. Since the ascription is based on a simulation of b's reasoning, this process is called *simulative belief ascription.* The origin set of the ascribed belief is computed by taking the union of the recorded ω_i for every $p_i \in \omega$. We are slightly sloppy here, because it is possible to have multiple source origin sets for a single p_i. In that case, we would need to generate all possible origin sets for the introduced belief and introduce it once for every different origin set. The I-flag of the result sentence is computed according to the I-flags of the used ω_i, but not according to the I-flag of the simulation result $!q$.

4.4.7 Examples

Now that we have completely defined the rules of $D_{\mathbf{SL}_0}$, we can look at a few examples to see how they can be used to model Cassie's reasoning. Figure 4.3 shows a simple example of Cassie's top-level reasoning. In steps 1 and 2 we introduce two base beliefs of Cassie as hypotheses. Again,

	Cassie ($\top$), I				
1	$!\mathsf{P}$,		hyp,	$\{!\mathsf{P}\}$	H
2	$!\mathsf{P} \Rightarrow \mathsf{Q}$,		hyp,	$\{!\mathsf{P} \Rightarrow \mathsf{Q}\}$	H
3	$!\mathsf{Q}$,		der,	$\{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}$	$\Rightarrow$E 1,2
4	$!\mathsf{B}(\mathsf{I}, \mathsf{Q})$,	I,	der,	$\{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}$	+IS 3
5	$!\mathsf{Q} \wedge \mathsf{B}(\mathsf{I}, \mathsf{Q})$,	I,	der,	$\{!\mathsf{P}, !\mathsf{P} \Rightarrow \mathsf{Q}\}$	$\wedge$I 3,4

Figure 4.3: Simple top-level reasoning in $\mathbf{SL}_0$

we use propositional constants such as P, Q, etc., to save space. In step 3 we derive a conclusion by applying the rule of implication-elimination (or modus ponens). In step 4 we apply the rule of positive introspection, and, since that rule encodes the reasoning agent of the context into its result, we add the I-flag to the left of the sentence. Finally, in step 5 we apply the rule of and-introduction. Note, that even though this was not an application of an introspective rule, the I-flag remains set, since we used a premise whose I-flag was set.

In Figure 4.4 we show a mirror image of the top-level reasoning described above, but now as a simulation of Mary's reasoning. Now the two initial beliefs are representations of Mary's beliefs. In steps 3 and 4 the object propositions of these beliefs are introduced into the Mary context as simulation hypotheses. Then a similar reasoning sequence as used in the previous example leads to

	Cassie ($\top$), I			
1	$!B(Mary, P)$,	hyp,	$\{!B(Mary, P)\}$	H
2	$!B(Mary, P \Rightarrow Q)$,	hyp,	$\{!B(Mary, P \Rightarrow Q)\}$	H
				open Mary
8	$!B(Mary, Q)$,	der,	$\{!B(Mary, P), !B(Mary, P \Rightarrow Q)\}$	BI 5
9	$!B(Mary, B(Mary, Q))$,	der,	$\{!B(Mary, P), !B(Mary, P \Rightarrow Q)\}$	BI 6
10	$!B(Mary, Q \wedge B(Mary, Q))$,	der,	$\{!B(Mary, P), !B(Mary, P \Rightarrow Q)\}$	BI 7

	Mary (Cassie), Mary, $!P/\{!B(Mary, P)\}, !P \Rightarrow Q/\{!B(Mary, P \Rightarrow Q)\}$				
3	$!P$,		hyp,	$\{!P\}$	SH 1
4	$!P \Rightarrow Q$,		hyp,	$\{!P \Rightarrow Q\}$	SH 2
5	$!Q$,		der,	$\{!P, !P \Rightarrow Q\}$	$\Rightarrow$E 3,4
6	$!B(Mary, Q)$,	I,	der,	$\{!P, !P \Rightarrow Q\}$	+IS 5
7	$!Q \wedge B(Mary, Q)$,	I,	der,	$\{!P, !P \Rightarrow Q\}$	$\wedge$I 5,6

Figure 4.4: Simple simulative reasoning in $\mathbf{SL}_0$

the results in steps 5, 6, and 7. The only difference is that the reasoning agent used by the rule of positive introspection is now Mary. Finally, these simulation results are ascribed to Mary in steps 8, 9, and 10 in Cassie's top-level context by applying the rule of belief introduction. Note how the origin sets of these sentences get computed by using the map stored at the top of the Mary context. Note also, that even though the sentences of steps 6 and 7 in the Mary context have the I-flag set, it does not get carried over to the corresponding sentences in steps 9 and 10 of the Cassie context. This is because the I-flag of such derived belief sentences is solely determined by the I-flags of the sentences in their origin sets; i.e., no mapping takes place across contexts as is the case for origin sets. For example, only if the sentences of steps 1 or 2 had their I-flag set would it also get set for the introduced belief sentences in steps 9 and 10.

Since writing full propositional terms in origin sets quickly leads to very crowded deductions, we will from now on use the step numbers of the lines on which they appeared as a sentence as names for them. For example, if the line with step number 5 contains the sentence $!P$, then we can use 5 as an alias for $!P$ in any origin set that contains it. Similarly, in the accompanying text, instead of saying "the sentence in step 5 is a hypothesis" we can abbreviate by saying "5 is a hypothesis". Since the same sentence can occur on multiple lines, names are not unique. If the uniqueness matters in a particular example, we will use the smallest step number in which the sentence appeared.

Figure 4.5 shows a simple two-level simulation in which Cassie simulates Mary's simulation of Sally's reasoning. Since the names of deduction contexts have to be unique, we name the context in which Mary simulates Sally's reasoning "MarySally". That way there can be other contexts in which other agents simulate Sally's reasoning without having a name conflict. Steps 3 to 6 import the innermost object propositions of the hypotheses in steps 1 and 2 all the way down into the Sally context. Then a simple one-step simulation derives the result in step 7. Finally, this result gets percolated to the top in steps 8 and 9 by applying the rule of belief introduction first in the Mary context and then in the Cassie context.

The example in Figure 4.6 shows how deductions in different contexts can be pursued in a

	Cassie (⊤), I			
1	!B(Mary, B(Sally, P)),	hyp,	{1}	H
2	!B(Mary, B(Sally, P ⇒ Q)),	hyp,	{2}	H
				open Mary
9	!B(Mary, B(Sally, Q)),	der,	{1, 2}	BI 8

	Mary (Cassie), Mary, 3/{1}, 4/{2}			
3	!B(Sally, P),	hyp,	{3}	SH 1
4	!B(Sally, P ⇒ Q),	hyp,	{4}	SH 2
				open MarySally
8	!B(Sally, Q),	der,	{3, 4}	BI 7

	MarySally (Mary), Sally, 5/{3}, 6/{4}			
5	!P,	hyp,	{5}	SH 3
6	!P ⇒ Q,	hyp,	{6}	SH 4
7	!Q,	der,	{5, 6}	⇒E 5,6

Figure 4.5: Two-level simulation in $\mathbf{SL}_0$

	Cassie (⊤), I			
				open CoCassie
1	!∀x Monster(x) ⇒ Ugly(x),	hyp,	{1}	H
3	!Monster(Herman),	hyp,	{3}	H
5	!Monster(Dracula),	hyp,	{5}	H
7	!Monster(Dracula) ⇒ Ugly(Dracula),	der,	{1}	∀E 1
9	!Ugly(Dracula),	der,	{1, 5}	⇒E 5,7
10	!Ugly(Herman),	der,	{1, 3}	⇒I 8
14	!Monster(Werewolf) ⇒ Ugly(Werewolf),	der,	{1}	⇒I 13

	CoCassie (Cassie), I			
2	!∀x Monster(x) ⇒ Ugly(x),	hyp,	{1}	R 1
4	!Monster(Herman),	hyp,	{3}	R 3
6	!Monster(Herman) ⇒ Ugly(Herman),	der,	{1}	∀E 2
8	!Ugly(Herman),	der,	{1, 3}	⇒E 4,6
11	!Monster(Werewolf),	hyp,	{11}	H
12	!Monster(Werewolf) ⇒ Ugly(Werewolf),	der,	{1}	∀E 2
13	!Ugly(Werewolf),	der,	{1, 11}	⇒E 11,12

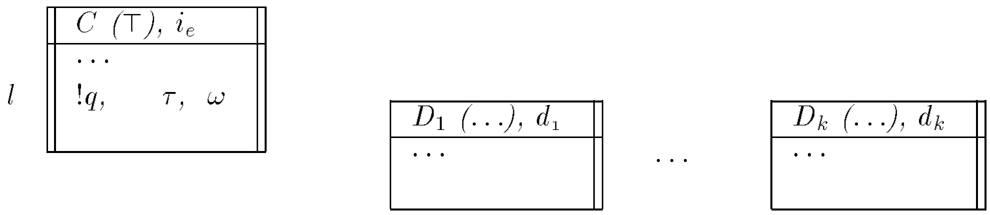
Figure 4.6: Pseudo-parallel reasoning in $\mathbf{SL}_0$

pseudo-parallel fashion, and how contexts can be reused at any point of a deduction. The opening of contexts is completely independent from any deductions already carried out. That is why we can open context CoCassie, even before the first deduction step takes place. The name CoCassie is meant to illustrate that the reasoning in it is interleaved with the reasoning in the main context in a coroutine-like fashion. After the initial two reiterations in context CoCassie the deductions that lead to the results in steps 8 and 9 could be pursued completely in parallel. This is illustrated by interleaving the steps in the way shown, since the rules of $D_{\mathbf{SL}_0}$ do force us to serialize deduction steps. In step 10, we export the result from step 8 into the Cassie context by way of implication introduction. Since at this point no hypotheses are raised in context CoCassie, the antecedents of the implication are empty. In step 11, we reuse CoCassie by raising an actual hypothesis which leads to the result in step 13. That then gets exported to context Cassie in step 14, this time leading to a proper implication. Of course, the result could have been achieved immediately by using universal elimination, but that was not the point of the example.

4.4.8 Derivability

Let us now define what it means for a sentence to be *derivable* from a set of hypotheses in the system $D_{\mathbf{SL}_0}$:

Definition 4.4.4. $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle, n \geq 0$ *iff there exists an* i_e*-deduction* Δ *of the form*



such that $hyps(C, l) \subseteq \{p_1, \ldots, p_n\}$*.*

Whenever it is clear from context what the various parameters are, we will use the shorthand $P \vdash !q$ to express the derivability relation ($P = \{!p_1, \ldots, !p_n\}$).

The connection between derivability and origin sets is expressed by the following theorem:

Theorem 4.4.1. *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle, n \geq 0$*, then* $\omega \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle$

This theorem tells us that the hypotheses recorded in the origin set of any sentence somewhere in a deduction context are actually sufficient to derive it in that context. Nothing else has to be assumed to be able to get it as a result. A proof for the theorem is given on page 167 in the Appendix. However, simply looking at the way origin sets get computed in the individual inference rules is enough to be convinced of its validity. Almost all inference rules compute a result origin set that is the union of the origin sets of the premises used. The only inference rules that produce smaller origin sets than that are the rules where hypotheses get discharged, i.e., ¬I, ⇒I, ∀I, ∃E, and BI, but it is easy to see how these rules still preserve a sufficient set of hypotheses to derive the result.

4.5 Soundness of $\mathbf{SL}_0$

The following soundness result shows that $D_{\mathbf{SL}_0}$ meshes well with the semantic account we presented in Section 4.3:

Theorem 4.5.1. (soundness): *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle, n \geq 0$*, and let* a *be any consistent agent interpretation structure* $\langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, int \rangle$ *such that* $[\![i_e]\!]_a = \boldsymbol{e}$ *and* $[\![q]\!]_a\downarrow$ *(we will call such interpretation structures* **admissible***), and where* $\models_a !p_1, \ldots, \models_a !p_n$*. Then* $\models_a !q$*.*

As a shorthand we will write this as $P \models !q$ with $P = \{!p_1, \ldots, !p_n\}$. Refer to page 169 in the Appendix for a proof of the theorem.

Chapter 5

Reasoning about Realistic Agents

Our primary motivation for developing a logic such as **SL** is to provide a formal foundation upon which the construction of an agent such as Cassie, or at least of parts of it, can be based. With this in mind, let us look closer at some aspects of Cassie's behavior as they are currently prescribed by the monotonic logic $\mathbf{SL}_0$ developed in the previous chapter. The main modeling assumption is that sentences of the language of $\mathbf{SL}_0$ represent propositions believed by Cassie. For example, Cassie's beliefs that Fido is a dog and that dogs are animals could be represented in her artificial mind by the sentences $!\mathsf{Dog}(\mathsf{Fido})$ and $!\forall \mathsf{x}\, \mathsf{Dog}(\mathsf{x}) \Rightarrow \mathsf{Animal}(\mathsf{x})$. Should she ever become interested to know whether Fido is an animal, an inference engine using the rules of the deductive system of $\mathbf{SL}_0$ would allow her to add the proposition $\mathsf{Animal}(\mathsf{Fido})$ to her set of believed propositions by deriving the sentence $!\mathsf{Animal}(\mathsf{Fido})$ and adding it to her artificial mind. Since $\mathbf{SL}_0$ is sound, we know that Cassie is actually justified in believing that Fido is an animal, and all is well.

If Cassie performs a similar reasoning sequence as a simulation of Mary's reasoning, she will arrive at the conclusion that Mary believes that Fido is an animal, or $!\mathsf{B}(\mathsf{Mary}, \mathsf{Animal}(\mathsf{Fido}))$, and she will hold that derived belief with the same conviction as the one in the previous example. Again, our semantic account tells us that Cassie is really justified in holding this new belief about Mary. This, however, is undesirable, because there is no way for Cassie to know whether Mary actually holds that belief or not. If Cassie would reason this way, then she would idealize other agents by assuming that they reason exactly as she does. We call this the undesirable property of *simulative idealization*, a characteristic that is somewhat reminiscent of the logical omniscience exhibited by modal logics of knowledge and belief. Logically omniscient agents believe everything within the infinite logical-consequence set of their base beliefs. Simulatively idealizing agents believe that other agents really believe everything ever inferred during simulations of the other agents' reasoning. Simulative idealization is not quite as dramatic an idealization as logical omniscience, but nevertheless it is undesirable.

To remedy these shortcomings, we extend $\mathbf{SL}_0$ into the full logic **SL**. The language of **SL** will be the same as it was for $\mathbf{SL}_0$, thus, $L_{\mathbf{SL}_0}$ equals $L_{\mathbf{SL}}$. The main difference between $\mathbf{SL}_0$ and **SL** lies in their deductive systems and in some of the semantics underlying belief.

5.1 Incomplete Agents

One property of realistic agents that we already alluded to above is that they are *incomplete*; that is, they do not believe everything which follows from their base beliefs. When Cassie simulates some

	Cassie ($\top$), I			
1	$!B(Oscar, \forall c_1, c_2, e\,(Class(c_1) \wedge Class(c_2) \wedge$			
	$\wedge Equiv(c_1, c_2) \wedge Member(e, c_2)) \Rightarrow Member(e, c_1)),$	hyp,	$\{1\}$	H
2	$!B(Oscar, Class(P)),$	hyp,	$\{2\}$	H
3	$!B(Oscar, Class(NP)),$	hyp,	$\{3\}$	H
4	$!B(Oscar, \forall p\, Member(p, P) \Rightarrow PolynomialTime(p)),$	hyp,	$\{4\}$	H
5	$!B(Oscar, Member(SAT, NP)),$	hyp,	$\{5\}$	H
				open Oscar
18	$!B(Oscar, Equiv(P, NP) \Rightarrow PolynomialTime(SAT)),$	der,	$\{1, 2, 3, 4, 5\}$	BI 17
	...			Exam
99	$!\neg B(Oscar, Equiv(P, NP) \Rightarrow PolynomialTime(SAT)),$	hyp,	$\{99\}$	H

	Oscar (Cassie), Oscar, $6/\{1\}, 7/\{2\}, 8/\{3\}, 9/\{4\}, 10/\{5\}$			
6	$!\forall c_1, c_2, e\,(Class(c_1) \wedge Class(c_2) \wedge Equiv(c_1, c_2) \wedge$			
	$\wedge Member(e, c_2)) \Rightarrow Member(e, c_1),$	hyp,	$\{6\}$	SH 1
7	$!Class(P),$	hyp,	$\{7\}$	SH 2
8	$!Class(NP),$	hyp,	$\{8\}$	SH 3
9	$!\forall p\, Member(p, P) \Rightarrow PolynomialTime(p),$	hyp,	$\{9\}$	SH 4
10	$!Member(SAT, NP),$	hyp,	$\{10\}$	SH 5
				open OscarHyp
17	$!Equiv(P, NP) \Rightarrow PolynomialTime(SAT),$	der,	$\{6, 7, 8, 9, 10\}$	$\Rightarrow$I 16

	OscarHyp (Oscar), Oscar			
11	$!Equiv(P, NP),$	hyp,	$\{11\}$	H
12	$!(Class(P) \wedge Class(NP) \wedge Equiv(P, NP) \wedge$			
	$\wedge Member(SAT, NP)) \Rightarrow Member(SAT, P),$	der,	$\{6\}$	$\forall$E 6
13	$!(Class(P) \wedge Class(NP) \wedge Equiv(P, NP) \wedge$			
	$\wedge Member(SAT, NP)),$	der,	$\{7, 8, 10, 11\}$	$\wedge$I 7,8,10,11
14	$!Member(SAT, P),$	der,	$\{6, 7, 8, 10, 11\}$	$\Rightarrow$E 12,13
15	$!Member(SAT, P) \Rightarrow PolynomialTime(SAT),$	der,	$\{9\}$	$\forall$E 9
16	$!PolynomialTime(SAT),$	der,	$\{6, 7, 8, 9, 10, 11\}$	$\Rightarrow$E 14,15

Figure 5.1: How simulative idealization can fail in a teaching situation

other agent's reasoning, she needs to take that into account, because a simulation result could be one of those conclusions that the simulated agent has not yet arrived at. A real-life example of such a situation is teaching. Many times a teacher teaches the basics of some subject and assumes that the "obvious" conclusions have been drawn by the students, only to find out later at an exam that that assumption was obviously wrong.

Figure 5.1 shows an example in which Cassie is imagined to be a teacher who teaches basic complexity theory. Oscar is one of Cassie's students of whom she assumes that from the material presented in class he has arrived at the following obvious (to her) conclusion: If the complexity classes P and NP are equivalent, then there exists an algorithm that solves any instance of the NP-complete satisfiability problem called SAT in polynomial time. Steps 1 to 5 in the top-level context show Cassie's beliefs about the state of Oscar's comprehension of the presented material. Step 1 represents her belief that he believes that, if two classes are equivalent, then every member of one class is also a member of the other. In addition to that, she believes that Oscar believes that P and NP are classes (steps 2 and 3), that for every member of P there is an algorithm that solves it in polynomial time (step 4), and that SAT is in NP (step 5).

The conclusion in step 18 is a result of simulative reasoning carried out in the simulation context Oscar and the hypothetical context OscarHyp. Steps 6 to 10 are simply the object propositions of the belief sentences of steps 1 to 5. Since Cassie has told her students many times that it is not known whether P equals NP, she simulates Oscar's reasoning as an instance of hypothetical reasoning. In step 11, she simulates his hypothetical assumption that P actually does equal NP. In step 12, she instantiates the universal sentence of step 6. The reiteration of 6 into the hypothetical context is omitted for brevity, since the sentences of steps 6 to 10 are all trivially visible in the hypothetical context. Step 13 combines four sentences into a single conjunction (the individual and-introduction steps are also omitted). In step 16, Cassie finally arrives at the conclusion that SAT is computable in polynomial time. Since that result uses the hypothesis of step 11, Cassie can introduce the entailment into the Oscar context in step 17 and then ascribe this simulation result to Oscar in step 18.

A few weeks later Cassie gives an exam. While she grades Oscar's exam she finds out — much to her dismay — that he obviously does not believe the proposition of step 18, otherwise he would have solved one of the exam problems correctly. This new belief of Cassie is represented in step 99, which directly contradicts the simulation result of step 18. So, what is she supposed to believe now?

If we do not take special action now, i.e., change the logic $\mathbf{SL}_0$ to take care of this case, then Cassie will be able (and justified) to derive and believe *any arbitrary sentence* by using contradiction-elimination based on the rule of negation-introduction. It is certainly completely undesirable to have Cassie's own top-level reasoning collapse just because one of the agents she knows about is incomplete.

There are actually two distinct scenarios in which the above contradiction could have arisen, but only the second one constitutes the case of an incomplete agent:

1. One or more of Cassie's initial beliefs about Oscar's beliefs are incorrect. This case can be handled by a standard belief revision operation where Cassie figures out which of these initial beliefs are actually incorrect and then retracts or *disbelieves* them. The various derived results based on one of the retracted beliefs will also be retracted. Automatically determining exactly what beliefs to retract is difficult, but, assuming that problem can be solved, the associated disbelief propagation is easily handled by a truth maintenance system (or, better, belief maintenance system) such as the one underlying the Multiple Belief Reasoner, MBR.

2. None of Cassie's initial beliefs about Oscar's beliefs are incorrect. For example, it is easily imaginable that each of these beliefs is directly "observable" by reading Oscar's exam paper, only Oscar's belief in the obvious conclusion is not manifested anywhere. Even worse, it is directly observable that he does not believe the conclusion in question. This case cannot be solved by belief revision, because there is nothing to revise. All the initial beliefs are correct and should not be retracted. The problem is that Oscar's reasoning is incomplete, and what needs to be done is to block the incorrect simulation result in light of the striking evidence to the contrary.

5.2 Treating Simulation Results as Default Conclusions

The above exposition makes clear that simulation results need to be treated as *default* conclusions rather than hard conclusions. Such simulation default conclusions can be shadowed if they contradict any belief that was acquired directly and is not itself based on a simulation also.

To handle the default character of simulation results at the logic level, we introduce the concept of a *simulation assumption.* A simulation assumption, or simply assumption, is a special kind of hypothesis that is justified by a derivation from a set of proper hypotheses. In a sense, an assumption is a hermaphrodite, because it is a hypothesis and a derived sentence simultaneously. This characterization of an assumption was introduced by Cravo and Martins [1993a; 1993b] in their formalization of default reasoning, and the following treatment owes a great deal to their work.

To take simulation assumptions into account, we have to extend Definition 4.4.1 for supported sentences (note again that $L_{\mathbf{SL}_0}$ equals $L_{\mathbf{SL}}$):

Definition 5.2.1. *A* **supported sentence** *is a quintuple $\langle !p, a/?, \tau, \omega, \alpha \rangle$, where (1) $!p \in S$ is an $L_{\mathbf{SL}}$ sentence, (2) $a/?$ is either some agent $a \in I$ or the unspecified agent ?, (3) τ is an origin tag which can be either* `hyp`*,* `der`*, or* `sim`*, (4) $\omega \subseteq S$ is the set of hypotheses, and (5) $\alpha \subseteq S$ is the set of simulation assumptions on which the derivation of $!p$ is based.*

The notation in a deduction context will also be slightly changed by adding the new assumption origin set to the right of the support of the sentence. For example, a possible contextual notation of the supported sentence $\langle !\mathsf{Q}, \mathsf{Mary}, \mathtt{der}, \{!\mathsf{S} \Rightarrow \mathsf{Q}\}, \{!\mathsf{S}\}\rangle$ is shown below:

	C (...), Mary
	...
m	$!\mathsf{Q}$, I, `der`, $\{!\mathsf{S} \Rightarrow \mathsf{Q}\}$, $\{!\mathsf{S}\}$

5.3 $D_{\mathbf{SL}}$: A Revised Version of $D_{\mathbf{SL}_0}$

Since we have introduced a new support element for supported sentences, we have to update the definitions of the various deduction rules of $D_{\mathbf{SL}_0}$. For most rules, the necessary changes are trivial; hence, we will only give the revised definitions of the rules that are substantially different from their previous version. This new set of deduction rules will constitute the deductive system $D_{\mathbf{SL}}$.

Hypothesis (H)

	C (...), a	
	...	
$l+1$	$!p$, `hyp`, $\{!p\}$, $\{\}$	H

Just as in $D_{\mathbf{SL}_0}$, the rule of hypothesis is used to add some new hypothesis $!p$ to a deduction context. The difference is that now the support of the new hypothesis has an additional empty set of simulation assumptions added to it.

And-Introduction ($\wedge$I)

$$
\begin{array}{r|llll|l}
 & C\ (\ldots),\ a & & & & \\
\hline
 & \ldots & & & & \\
m & !p, & \tau_1, & \omega_1,\ \alpha_1 & & \\
 & \ldots & & & & \\
n & !q, & \tau_2, & \omega_2,\ \alpha_2 & & \\
 & \ldots & & & & \\
l+1 & !p \wedge q, & \mathtt{der}, & \omega_1 \cup \omega_2,\ \alpha_1 \cup \alpha_2 & & \wedge\mathrm{I}\quad m, n \\
l+1 & !q \wedge p, & \mathtt{der}, & \omega_1 \cup \omega_2,\ \alpha_1 \cup \alpha_2 & & \wedge\mathrm{I}\quad m, n
\end{array}
$$

This rule serves as a prototypical example of the various standard rules not repeated in this section. The only difference to the $D_{\mathbf{SL}_0}$ version of the individual rules is that now operations on origin sets are always performed separately on the hypothesis and the assumption origin set.

Negation-Introduction ($\neg$I)

$$
\begin{array}{r|lll|l}
 & C\ (\ldots),\ a & & & \\
\hline
 & \ldots & & & \\
m & !p \wedge \neg p, & \tau, & \omega \cup \{!h\},\ \{\} & \\
 & \ldots & & & \\
l+1 & !\neg h, & \mathtt{der}, & \omega \setminus \{!h\},\ \{\} & \neg\mathrm{I}\quad m
\end{array}
$$

From a contradiction that is solely based on hypotheses, we can deduce the negation of any hypothesis on which the derivation of the contradiction was based. Following Cravo and Martins, we will call such a contradiction a *real* contradiction, as opposed to an *apparent* contradiction, which is partly based on assumptions.

Simulation-Hypothesis (SH)

$$
\begin{array}{r|lll|}
 & C\ (\ldots),\ a & & \\
\hline
 & \ldots & & \\
m & !\mathsf{B}(b, p_i), & \tau, & \omega_i,\ \{\} \\
 & \ldots & &
\end{array}
$$

$$
\begin{array}{r||lll||l}
 & D\ (C),\ b, & & & \\
 & !p_1/\omega_1\alpha_1, \ldots, !p_i/\omega_i\{\} & & & \\
\hline
 & \ldots & & & \\
l+1 & !p_i, & \mathtt{hyp}, & \{!p_i\},\ \{\} & \mathrm{SH}\quad m
\end{array}
$$

The rule of simulation hypothesis now comes in two variants depending on whether the parent belief sentence is based on any assumptions or not. If it does not depend on any assumptions, then the rule works just like before with the only difference (similar to the new version of the rule of hypothesis) that we add an empty assumption origin set to the support of the new simulation hypothesis. Since supports now can also contain assumptions, we have to record them in the origin set mapping in the top field of the simulation context.

Simulation-Hypothesis′ (SH′)

$$
\begin{array}{r|l||}
\cline{2-2}
 & C\ (\ldots),\ a \\
\cline{2-2}
 & \ldots \\
m & !\mathsf{B}(b, p_i),\ \ \tau,\ \ \omega_i,\ \alpha_i \neq \emptyset \\
 & \ldots \\
 & \\
\cline{2-2}
\end{array}
$$

$$
\begin{array}{r||l||l}
\cline{2-2}
 & D\ (C),\ b, & \\
 & !p_1/\omega_1\alpha_1, \ldots, !p_i/\omega_i\alpha_i & \\
\cline{2-2}
 & \ldots & \\
l+1 & !p_i, \qquad \mathtt{sim},\ \ \{\},\ \{!p_i\} & \mathrm{SH}'\ \ m \\
 & & \\
\cline{2-2}
\end{array}
$$

If the parent belief sentence does depend on some other simulation assumptions, we automatically make it into a simulation assumption in the simulation context. This makes sure that the policy "once an assumption always an assumption" stays enforced, because we do not want to promote the object proposition of an assumption-dependent belief sentence into a hypothesis once it gets used in a simulation context.

Belief-Introduction (BI)

$$
\begin{array}{r|l||l}
\cline{2-2}
 & C\ (\ldots),\ a & \\
\cline{2-2}
 & \ldots & \\
l+1 & !\mathsf{B}(b, q),\ \ \mathtt{sim},\ \ \displaystyle\bigcup_{!p_i \in \omega \cup \alpha} \omega_i,\ \{!\mathsf{B}(b, q)\} \cup \bigcup_{!p_i \in \omega \cup \alpha} \alpha_i & \mathsf{BI}\ \ m \\
 & & \\
\cline{2-2}
\end{array}
$$

$$
\begin{array}{r||l||}
\cline{2-2}
 & D\ (C),\ b, \\
 & !p_1/\omega_1\alpha_1, \ldots, !p_n/\omega_n\alpha_n \\
\cline{2-2}
 & \ldots \\
m & !q, \qquad \tau,\ \ \omega \subseteq shyps(D, m),\ \alpha \\
 & \ldots \\
 & \\
\cline{2-2}
\end{array}
$$

Belief-introduction is the only rule of $D_{\mathbf{SL}}$ that actually derives simulation assumptions. Note that the support of a new simulation assumption is basically the same as was computed in the $D_{\mathbf{SL}_0}$ version of the rule; the main difference is that the new assumption additionally to its standard $D_{\mathbf{SL}_0}$ origin set now also contains itself in its assumption origin set. Assumptions recorded in α which might have accumulated during nested simulations do not "survive" during the export to the parent context, since no mapping gets recorded for them. Again, we are slightly sloppy here, because such assumptions might be rederivations of assumptions introduced by the rule SH′. To handle proper support computation of such multiple derivations, we need a more complicated mapping mechanism. However, to keep things simple in **SL**-derivations, we will simply assume that such cases will not occur. In the implementation of SIMBA on the other hand, a more general scheme is used which can gracefully handle multiple derivations.

The standard part of the origin set of the derived belief sentence constitutes the justification for raising this particular assumption, since its elements (hypotheses as well as assumptions) made the derivation of the new simulation result possible. Adding the simulation assumption to its own assumption origin set makes it into a double-natured entity called a *justified hypothesis*. Its hypothesis nature has the additional benefit that all derived results based on it will record their dependency on this particular simulation assumption in their assumption origin set.

5.3.1 Example

Figure 5.2 shows an example application of the new deduction rules. Note how the result in step

	Cassie ($\top$), I			
1	$!\mathsf{B}(\mathsf{Mary}, \mathsf{P})$,	hyp,	$\{1\}, \{\}$	H
2	$!\mathsf{B}(\mathsf{Mary}, \mathsf{P} \Rightarrow \mathsf{Q})$,	hyp,	$\{2\}, \{\}$	H
3	$!\mathsf{B}(\mathsf{Mary}, \mathsf{Q}) \Rightarrow \mathsf{B}(\mathsf{Sally}, \mathsf{R})$,	hyp,	$\{3\}, \{\}$	H
4	$!\mathsf{B}(\mathsf{Sally}, \mathsf{R} \Rightarrow \mathsf{S})$,	hyp,	$\{4\}, \{\}$	H
				open Mary
8	$!\mathsf{B}(\mathsf{Mary}, \mathsf{Q})$,	sim,	$\{1, 2\}, \{8\}$	BI 7
9	$!\mathsf{B}(\mathsf{Sally}, \mathsf{R})$,	der,	$\{1, 2, 3\}, \{8\}$	$\Rightarrow$E 3,8
				open Sally
13	$!\mathsf{B}(\mathsf{Sally}, \mathsf{S})$,	sim,	$\{1, 2, 3, 4\}, \{8, 13\}$	BI 12

	Mary (Cassie), Mary, $5/\{1\}\{\}, 6/\{2\}\{\}$			
5	$!\mathsf{P}$,	hyp,	$\{5\}, \{\}$	SH 1
6	$!\mathsf{P} \Rightarrow \mathsf{Q}$,	hyp,	$\{6\}, \{\}$	SH 2
7	$!\mathsf{Q}$,	der,	$\{5, 6\}, \{\}$	$\Rightarrow$E 5,6

	Sally (Cassie), Sally, $10/\{1, 2, 3\}\{8\}, 11/\{4\}\{\}$			
10	$!\mathsf{R}$,	sim,	$\{\}, \{10\}$	SH$'$ 9
11	$!\mathsf{R} \Rightarrow \mathsf{S}$,	hyp,	$\{11\}, \{\}$	SH 4
12	$!\mathsf{S}$,	der,	$\{11\}, \{10\}$	$\Rightarrow$E 10,11

Figure 5.2: A simple simulation using the rules of $D_{\mathbf{SL}}$

13 contains an additional simulation assumption in its origin set, because the simulation of Sally's reasoning was partly based on a simulation of Mary's reasoning.

5.4 Believability

The only deduction rule of the new deductive system $D_{\mathbf{SL}}$ that actually changes the number of sentences derivable from a particular set of premises is the new version of negation-introduction. Without that rule, $D_{\mathbf{SL}_0}$ and $D_{\mathbf{SL}}$ would be identical in derivational power (the proof of that is lengthy but rather straightforward). What is changed by the new variant of negation introduction is that contradictions lurking within the object propositions of belief sentences that get uncovered by simulative reasoning are not considered real contradictions anymore; hence, negation introduction will be applicable in fewer cases than before. However, this reduction of derivable sentences does not yet solve the problem of what Cassie is supposed to believe if a simulation result contradicts a belief sentence that is solely based on proper hypotheses.

For example, imagine that, in the example in Figure 5.2, Sally tells Cassie that she does not believe S, and Cassie adds $\langle !\neg\mathsf{B}(\mathsf{Sally},\mathsf{S}), ?, \mathtt{hyp}, \{\neg\mathsf{B}(\mathsf{Sally},\mathsf{S})\}, \{\}\rangle$ to her top-level context. This new hypothesis directly contradicts the simulation result in step 13, but there is nothing that prevents the two sentences from peaceful coexistence. Even worse, Cassie could still derive $!\mathsf{B}(\mathsf{Sally},\mathsf{S}) \wedge \neg\mathsf{B}(\mathsf{Sally},\mathsf{S})$; the only thing she could not do is to apply the rule of negation-introduction to that contradiction.

The problem is that in the monotonic system $D_{\mathbf{SL}_0}$ *derivability* was equivalent to *believability*. That is, if Cassie was justified in believing a set of premises, then she was also justified in believing everything derivable from these premises. In $D_{\mathbf{SL}}$, derivability and believability are not equivalent any more, and we have to define a more restrictive notion of believability based on the weaker or more permissive notion of derivability.

5.4.1 Belief States

During the "lifetime" of Cassie, her deduction contexts change. New sentences are derived or get added as hypotheses, obsolete hypotheses get removed or disbelieved, and what is perfectly believable at one point might not be believable any more sometime later. Said differently, the set of propositions believed by Cassie does not grow monotonically, which is the main reason why we need a nonmonotonic logic to formalize her reasoning.

To get a handle on the changing set of believable propositions in a deduction context, we will look at individual snapshots of such a context. These snapshots will be called *belief states*, and they are defined as follows:[1]

Definition 5.4.1. *A* **belief state** σ *is a quadruple* $\langle\langle a, H, A, \sigma_p\rangle\rangle$ *where*

1. $a \in I$ *is a reasoning agent,*

2. $H \subseteq S$ *is a set of sentences taken to be hypotheses,*

3. $A \subseteq S$ *is a set of sentences taken to be a priori simulation assumptions, and*

4. σ_p *is either* $\top$ *or another belief state taken to be the parent or simulator belief state.*

Below, we will define what believability relative to a belief state means. Intuitively, we will say that if the agent of a belief state believes exactly the set of propositions specified by the belief state, then only propositions in a particular subset of the deductive closure of the belief state will be believable by the agent.

5.4.2 $\mathbf{SL}_0$ Believability

For the monotonic logic $\mathbf{SL}_0$, believability can be defined as follows:

Definition 5.4.2. *A supported sentence* $\langle !p, a_s, \tau, \omega\rangle$ *renders its proposition* p **believable** *in a belief state* $\langle\langle a_\sigma, H, \{\}, \top\rangle\rangle$ *iff the following conditions hold:*

1. *Agents* a_s *and* a_σ *are compatible; i.e., they are either identical or* a_s *is the arbitrary agent* $?$.

2. $\omega \subseteq H$.

[1] Cravo and Martins call the concept corresponding to a belief state a *context*.

Believability of a proposition in a belief state as defined above is always determined with respect to a particular support. One support might render it believable; another support might render it as unbelievable. This is the case, since the above definition does not prescribe any connection between the belief state and the supported sentence. However, we usually do have such a connection, as exhibited by the corollary below.

Because of Theorem 4.4.1, the following holds:

Corollary 5.4.1. *If* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, i_e, \tau, \omega \rangle$, *then the proposition* q *is believable in belief state* $\langle\langle i_e, \{!p_1, \ldots, !p_n\}, \{\}, \sigma_p \rangle\rangle$.

The corollary says that, in the monotonic case, whatever is derivable is also believable, which is the reason why we did not give a definition of believability then.

5.4.3 SL-Derivability

For the following exposition, it is advantageous to slightly depart from the format of Definition 4.4.4 and define **SL**-derivability with help of a belief state:

Definition 5.4.3. $\langle\langle a, H, A, \sigma_p \rangle\rangle \vdash_{a, D_{\mathbf{SL}}} \langle !q, a/?, \tau, \omega, \alpha \rangle$ *iff there exists a deduction* Δ *of the form*

$$\begin{array}{cccc} \boxed{\begin{array}{l} C\ (\top),\ i_e \\ \hline \ldots \\ \\ \end{array}} & \ldots & l\ \boxed{\begin{array}{l} D\ (\ldots),\ a \\ \hline \ldots \\ !q, \quad \tau,\ \omega,\ \alpha \\ \\ \end{array}} & \ldots \end{array}$$

such that $hyps(D, l) \subseteq H$ *and* $shyps'(D, l) \subseteq A$.

As before, we take $hyps(D, l)$ as the set of all sentences introduced with the rules H or SH. The new function $shyps'$ collects all sentences introduced with the new rule SH$'$. This definition of derivability is applicable to belief states describing top-level contexts as well as nested simulation contexts. Since a belief state already specifies a reasoning agent, there is usually no need to also parameterize the derivation turnstile with that agent.

Similar to Theorem 4.4.1, the following holds:

Theorem 5.4.1. *If* $\langle\langle a, H, A, \sigma_p \rangle\rangle \vdash \langle !q, a/?, \tau, \omega, \alpha \rangle$, *then* $\langle\langle a, \omega, A \cap \alpha, \sigma_p \rangle\rangle \vdash \langle !q, a/?, \tau, \omega, \alpha \rangle$.

Often it is more convenient to state derivability in terms of a deductive closure:

Definition 5.4.4. *Let* $\sigma = \langle\langle a, H, A, \sigma_p \rangle\rangle$. *The* **deductive closure** *of* σ *is the set*

$$\sigma^\star = \{\langle !q, a/?, \tau, \omega, \alpha \rangle \mid \sigma \vdash \langle !q, a/?, \tau, \omega, \alpha \rangle\}$$

5.4.4 SL-Believability

The **SL** definition of believability should look something like the following:

Definition 5.4.5. *A supported sentence* $\langle !p, a_s, \tau, \omega, \alpha \rangle$ *renders its proposition* p **believable** *in a belief state* $\langle\langle a_\sigma, H, A, \sigma_p \rangle\rangle$ *iff the following conditions hold:*

1. Agents a_s and a_σ are compatible.

2. $\omega \subseteq H$.

3. α is contained in a set of "reasonable simulation assumptions" determined by the belief state.

The problem remaining is to specify what these reasonable simulation assumptions really are. Here are some constraints for an assumption to be reasonable relative to a belief state:

1. It should be motivated by the hypotheses of the belief state; i.e., it should be derivable from these hypotheses.

2. It should not contradict any of the hypotheses or any of their sound consequences (sound consequences are those not dependent on simulation assumptions).

3. It should not contradict any of the other reasonable assumptions motivated by the belief state.

Using standard default logic terminology (e.g., see [Reiter, 1980; Besnard, 1989]), such a set of reasonable assumptions constitutes an *extension* that can be used to complete an incomplete theory such as the agent theory defined by a belief state.

Formal treatments of default reasoning have to deal with two main problems which we already partly alluded to in the above characterization of reasonability:

1. Defining the set of default conclusions or assumptions that are consistent with the underlying base theory.

2. Properly handling mutually inconsistent default assumptions, or the problem of multiple extensions.

To do that, an extension of a default theory $\Delta = (D, W)$ is typically defined in two steps: First, a function Γ is defined which, when applied to a closed set of sentences S, yields the smallest deductively closed superset of the base theory W that contains all default conclusions motivated by the defaults D that are individually consistent with S. Second, an extension E of Δ is defined as a fixed point of the function Γ; i.e., E is a closed set of sentences for which $\Gamma(E) = E$ (cf. [Reiter, 1980, p. 89]).

The notion of fixed point is theoretically elegant, but fixed points are hard to work with in practice. It is difficult to get a good intuitive grasp on what fixed points look like; they cannot be computed, since they are infinite sets; and it is usually also hard to describe how to approximate them. For that reason, we adapt the approach of Cravo and Martins for our purposes. Their approach has the appeal of being conceptually simpler, and it will allow us to define extensions iteratively.

The main difference between the default character of **SL** and that of a default logic such as Reiter's or Cravo's is that **SL** does not contain any individual default rules such as, for example, "birds usually fly", which in Reiter's notation for defaults would be expressed as:

$$\frac{\mathsf{Bird(x)} : \mathbf{M}\mathsf{Fly(x)}}{\mathsf{Fly(x)}}$$

The above default rule is to be read as "if x is a bird, and it is consistent to assume that x can fly, then infer that x can fly". In **SL**, instead of individual default rules, we have a general schema which states that, by default, simulation results are actually believed by the simulated agent. Somewhat abusing Reiter's default notation, that schema could be expressed by the following "default rule":

$$\frac{\sigma_b \vdash p : \mathsf{MB}(b, p)}{\mathsf{B}(b, p)}$$

This rule might be paraphrased as "if the simulation of some agent b's reasoning in some belief state σ_b leads to the conclusion p, and it is consistent in its parent belief state to assume that b believes p, then assume that b believes p".

5.4.5 Degrees of SL-Believability

Here is a short outline of what is about to follow:

1. Given a belief state $\sigma = \langle\langle a_\sigma, H, A, \sigma_p \rangle\rangle$, we will define its *prima facie extension* $P(\sigma)$ as the set of supported simulation assumptions that each individually is justified by and consistent with belief state σ.

2. The prima facie extension of σ will then be partitioned into a set $\mathcal{E}(\sigma)$ of (proper) extensions, where each extension $E \in \mathcal{E}(\sigma)$ is a maximal subset of $P(\sigma)$ that is deductively closed and consistent with σ.[2]

Definition 5.4.6. *Let* $\sigma = \langle\langle a, H, A, \sigma_p \rangle\rangle$ *be a belief state. An* **extension set** *for* σ *is any set* E *such that* $E \subseteq \{\langle !p, a/?, \mathtt{sim}, \omega, \alpha\rangle \in \sigma^\star \mid \alpha \neq \emptyset\}$.

Thus, any set of simulation assumptions derivable from a belief state can be an extension set. Note, that the a priori assumptions A are trivially derivable from σ.

Definition 5.4.7. *Let* $\sigma = \langle\langle a_\sigma, H, A, \sigma_p \rangle\rangle$ *be a belief state and* E *an extension set for* σ. *A supported sentence* $\langle !p, a_s/?, \tau, \omega, \alpha\rangle$ *renders its proposition* p **believable** *in* σ *extended by* E *iff:*

1. a_σ *and* $a_s/?$ *are compatible, and*

2. $\omega \subseteq H$, *and*

3. $\alpha \subseteq sent(E) \cup H$ *('sent' is a projection function that selects the plain sentences from a set of supported sentences such as* E*).*

For a sentence based on assumptions, the least we require to render believability to its proposition in a belief state σ is that its hypothesis origin set be contained in H and that its assumption origin set be contained in the extension set E used to extend σ. If some of these assumptions are actually hypotheses of σ rather than elements of E, then that is even better. This possibility is expressed by including the set of hypotheses in the last condition of the above definition. Because the believability of propositions might depend on a priori assumptions, the set A has to be taken into account in the construction of extensions.

Regardless of what the *correct* extensions of a belief state will turn out to be, we already can define the following *degrees of believability* of a proposition relative to a belief state and a set of arbitrary extension sets:

[2] Cravo and Martins use the concepts *extended context* and *core* instead of prima facie and proper extensions.

Definition 5.4.8. *Let $\sigma = \langle\!\langle a_\sigma, H, A, \sigma_p \rangle\!\rangle$ be a belief state and $\mathcal{A}(\sigma)$ a set of arbitrary extension sets. A supported sentence $\langle !p, a_s/?, \tau, \omega, \alpha \rangle$ renders its proposition p*

- **certain**, *written with a bold exclamation mark $\mathbf{!}p$, iff its support renders it believable in belief state $\sigma = \langle\!\langle a_\sigma, H, \emptyset, \sigma_p \rangle\!\rangle$,*

- **plausible**, *written $\overset{\sim}{!}p$, iff either it is certain, or $\mathcal{A}(\sigma) \neq \emptyset$ and for every $E \in \mathcal{A}(\sigma)$ its support renders it believable in σ extended by E,*

- **possible**, *written $\overset{\approx}{!}p$, iff either it is plausible, or $\mathcal{A}(\sigma) \neq \emptyset$ and for at least one $E \in \mathcal{A}(\sigma)$ its support renders it believable in σ extended by E,*

- *or* **unbelievable**, *written $\circledast p$, if its support does not render it possible.*

The new symbols $\mathbf{!}$ (for "certain"), $\overset{\sim}{!}$ (for "approximate"), $\overset{\approx}{!}$ (for "very approximate"), and $\circledast$ (for "out") are not part of the language of **SL**, but are so far merely annotations to indicate the degree of believability of a particular proposition. How the different believabilities are to affect Cassie's reasoning still needs to be defined.

Similar to plain believability, a particular degree of believability of a proposition in a belief state is always connected to a particular support. One support might render a proposition certain, while a different support might render it only plausible. This is perfectly reasonable, since a support summarizes a derivation, and while one derivation of a proposition could have been based on hypotheses only, another might have involved assumptions. Of course, if a proposition has multiple supports, one will usually characterize it according to its strongest support. Similarly, since the degrees defined above include each other, e.g., every *plausible* proposition is also *possible*, we will always annotate it according to the strongest degree determined by a particular support.

Here is a short summary of the various different notations used with propositions and what they mean:

- P denotes some particular though unspecified proposition, for example, Lucy is sweet.

- $!\mathsf{P}$ means the agent whose mind contains this sentence (usually taken to be Cassie) believes the proposition denoted by P.

- $\langle !\mathsf{P}, ?, \mathtt{der}, \{!\mathsf{Q}, !\mathsf{R}\}, \{\} \rangle$ means the agent whose mind contains $!\mathsf{P}$ believes the proposition denoted by P, and its reason for believing it is that its mind also contains $!\mathsf{Q}$ and $!\mathsf{R}$.

- $\overset{\sim}{!}\mathsf{P}$ means the agent whose mind contains $!\mathsf{P}$ believes the proposition denoted by P, and that its associated support renders it *plausible* relative to the current belief state of the agent.

Having made these important differences explicit gives us the license to be sloppy from now on wherever the context allows. For example, we will often talk about the believabilities of *sentences*, even though believability is a property of the *proposition* of a sentence.

In the examples to follow, the believabilities of propositions will always be shown relative to a set of extension sets $\mathcal{A}(\sigma)$. The individual definitions of $\mathcal{A}(\sigma)$ will vary as we refine our definition of prima facie extensions and extensions.

5.4.6 One-Level Extensions

For matters of exposition we will first define extensions for a one-level reasoner and then in a second step remove that restriction. A one-level reasoner is an agent who only performs simulations one level deep, i.e., it simulates the reasoning of agents it directly knows about, but not their simulation of other agents.

The informal introduction above already showed that the definition of extensions critically hinges on the notion of *consistency*, which we now properly define:

Definition 5.4.9. *A belief state* $\langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$ *is* **consistent** *iff there is no* $!p \in S$ *such that* $\langle\!\langle a, H, A, \sigma_p \rangle\!\rangle \vdash\!\!- \langle !p \wedge \neg p, a/?, \tau, \omega, \{\} \rangle$. *It is* **strongly consistent** *iff there is no* $!p \in S$ *such that* $\langle\!\langle a, H, A, \sigma_p \rangle\!\rangle \vdash\!\!- \langle !p \wedge \neg p, a/?, \tau, \omega, \alpha \rangle$.

Plain consistency only requires that none of the sound consequences of the belief state is a contradiction. The sound consequences are those solely based on hypotheses. Strong consistency additionally requires that none of the consequences based on one or more simulation assumptions is a contradiction.

With that available, we are ready to define one-level prima facie extensions:

Definition 5.4.10. *Let* $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$. *Its* **one-level prima facie extension** $P_1(\sigma)$ *is:*

$$P_1(\sigma) = \{\langle !p, a/?, \mathtt{sim}, \omega, \alpha \rangle \in \sigma^\star \mid \langle\!\langle a, H \cup \alpha, \{\}, \sigma_p \rangle\!\rangle \textit{ is consistent}\}$$

$P_1(\sigma)$ is the set of all simulation assumptions derivable from σ whose assumption origin set is consistent with the hypotheses of the belief state in the following sense: If we added the sentences of the assumption origin set of such a simulation assumption as proper hypotheses to σ, we would still wind up with a consistent belief state. Note, that only consistency between each individual assumption and the hypotheses of σ is required, but nothing is said about consistency between assumptions. This is why we call it a *prima facie* extension, since prima facie its elements could form a proper extension. Moreover, only plain consistency of the extended belief state $\langle\!\langle a, H \cup \alpha, \{\}, \sigma_p \rangle\!\rangle$ is required, since if it was possible to derive contradictions based on assumptions starting from σ, then these apparent contradictions will be derivable again from the extended belief state.

To be precise, in the above definition "one-level" really means that the prima facie extension is defined by only looking at sentences at the top level of the deduction context described by the belief state. The derivations that led to these sentences, however, might have involved simulations at arbitrary depth. This will hopefully become clear shortly.

Corollary 5.4.2. *If a belief state* $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$ *is inconsistent, then* $P_1(\sigma) = \emptyset$.

Obviously, if we have an inconsistent belief state σ to begin with, every simulation assumption added to it will lead to another inconsistent belief state, and, hence, the prima facie extension of σ will be empty. Since only consistent belief states lead to interesting extensions, we will from now on always assume that the belief states we are considering are consistent.

Figure 5.3 is a variation of the example in Figure 5.2 that shows how some simulation results can be part of $P_1(\sigma)$ while others cannot. The top-level deduction context is described by two consecutive belief states σ_1 and σ_2. Sentence $!\mathsf{B}(\mathsf{Sally}, \mathsf{S})$ can be an element of the prima facie extension of σ_1, but once we add the new hypothesis in step 99, it can no longer be part of the prima facie extension of the new belief state σ_2. The believabilities are shown according to σ_2 and

	Cassie (⊤), I			
1	!B(Mary, P),	hyp,	{1}, {}	H
2	!B(Mary, P ⇒ Q),	hyp,	{2}, {}	H
3	!B(Mary, Q) ⇒ B(Sally, R),	hyp,	{3}, {}	H
4	!B(Sally, R ⇒ S),	hyp,	{4}, {}	H
				open Mary
8	!B(Mary, Q),	sim,	{1, 2}, {8}	BI 7
9	!B(Sally, R),	der,	{1, 2, 3}, {8}	⇒E 3,8
				open Sally
13	!B(Sally, S),	sim,	{1, 2, 3, 4}, {8, 13}	BI 12
	...			
99	!¬B(Sally, S),	hyp,	{99}, {}	H

	Mary (Cassie), Mary, 5/{1}{}, 6/{2}{}			
5	!P,	hyp,	{5}, {}	SH 1
6	!P ⇒ Q,	hyp,	{6}, {}	SH 2
7	!Q,	der,	{5, 6}, {}	⇒E 5,6

	Sally (Cassie), Sally, 10/{1, 2, 3}{8}, 11/{4}{}			
10	!R,	sim,	{}, {10}	SH′ 9
11	!R ⇒ S,	hyp,	{11}, {}	SH 4
12	!S,	der,	{11}, {10}	⇒E 10,11

$\sigma_1 = \langle\!\langle \mathsf{I}, \{1,2,3,4\}, \{\}, \top \rangle\!\rangle$
$P_1(\sigma_1) = \{\langle !\mathsf{B}(\mathsf{Mary},\mathsf{Q}), ?, \mathtt{sim}, \{1,2\}, \{8\}\rangle, \langle !\mathsf{B}(\mathsf{Sally},\mathsf{S}), ?, \mathtt{sim}, \{1,2,3,4\}, \{8,13\}\rangle, \ldots\}$

$\sigma_2 = \langle\!\langle \mathsf{I}, \{1,2,3,4,99\}, \{\}, \top \rangle\!\rangle$
$P_1(\sigma_2) = \{\langle !\mathsf{B}(\mathsf{Mary},\mathsf{Q}), ?, \mathtt{sim}, \{1,2\}, \{8\}\rangle, \ldots\}$
$\mathcal{A}(\sigma_2) =_{def} \{P_1(\sigma_2)\}$

Figure 5.3: A simple simulation with one-level prima facie extensions

a set of extension sets $\mathcal{A}(\sigma_2)$, which is defined as containing $P_1(\sigma_2)$ as its only element. The prima facie extensions are infinite sets which is indicated by the dots.

To be thorough, in order to *know* whether sentences 9 and 13 really are elements of the shown prima facie extensions, we would need to prove that they can be consistently added to the hypotheses of the respective belief states. Such proof is generally not only difficult, but, more importantly, an at best semidecidable problem, since it is at least as difficult as proving whether a contradiction is derivable from the resulting theory in a first-order logic. This is an important observation which in the implementation of an actual reasoning engine will lead us to settle for the weaker notion of “not known to be inconsistent”, instead of the stronger but impossible to compute notion of “known to be consistent”.

Figure 5.4 shows how unbelievability propagates. The simulation result !B(Mary, Q) in step 9

	Cassie (⊤), I			
1	!B(Mary, P),	hyp,	{1}, {}	H
2	!B(Mary, P ⇒ Q),	hyp,	{2}, {}	H
3	!¬B(Mary, Q),	hyp,	{3}, {}	H
4	!B(Mary, Q) ⇒ B(Sally, R),	hyp,	{4}, {}	H
5	!B(Sally, R ⇒ S),	hyp,	{5}, {}	H
				open Mary
9	$\textcircled{!}$B(Mary, Q),	sim,	{1, 2}, {9}	BI 8
10	$\textcircled{!}$B(Sally, R),	der,	{1, 2, 4}, {9}	⇒E 4,9
				open Sally
14	$\textcircled{!}$B(Sally, S),	sim,	{1, 2, 4, 5}, {9, 14}	BI 13

	Mary (Cassie), Mary, 6/{1}{}, 7/{2}{}			
6	!P,	hyp,	{6}, {}	SH 1
7	!P ⇒ Q,	hyp,	{7}, {}	SH 2
8	!Q,	der,	{6, 7}, {}	⇒E 6,7

	Sally (Cassie), Sally, 11/{1, 2, 4}{9}, 12/{5}{}			
11	!R,	sim,	{}, {11}	SH′ 10
12	!R ⇒ S,	hyp,	{12}, {}	SH 5
13	!S,	der,	{12}, {11}	⇒E 11,12

$\sigma = \langle\!\langle \mathsf{I}, \{1,2,3,4,5\}, \{\}, \top \rangle\!\rangle$
$P_1(\sigma) = \{\ldots\}$
$\mathcal{A}(\sigma) =_{def} \{P_1(\sigma)\}$

Figure 5.4: Simulation with propagated unbelievability

is not part of the one-level prima facie extension and, hence, is unbelievable. Because of that, !B(Sally, R) and !B(Sally, S), whose supports are both based on it, are also rendered unbelievable. Moreover, the simulation assumption !B(Sally, S) is not part of $P_1(\sigma)$ either, since its origin set contains a sentence that is not consistent with the hypotheses of σ.

Figure 5.5 shows an example where a prima facie extension is not also a satisfactory proper extension. Again, the believabilities are shown as if it would be by defining $\mathcal{A}(\sigma)$ as the singleton set containing $P_1(\sigma)$, but that leaves !S ∧ ¬S in step 15 a plausibly believable sentence, which is of course highly undesirable. The problem is that the simulation assumptions !B(Mary, Q) and !B(Mary, R) are each individually consistent with σ, but not when taken together. To deal with this problem of mutually inconsistent simulation assumptions, we define the (proper) extensions of σ by partitioning its prima facie extension into maximal consistent subsets.

Definition 5.4.11. *Let S, T be arbitrary sets and P a predicate. S is a* **maximal subset** *of T such that $P(S)$ iff $S \subseteq T$ and $P(S)$ is true and for any $e \in (T \setminus S)$ $P(S \cup \{e\})$ is false.*

Definition 5.4.12. *Let $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$. $\mathcal{E}_1(\sigma) = \{E \subseteq P_1(\sigma)\}$ is the set of* **one-level extensions** *of σ iff each E is a maximal subset of $P_1(\sigma)$ such that:*

	Cassie ($\top$), I			
1	! B(Mary, P),	hyp,	$\{1\}, \{\}$	H
2	! B(Mary, P $\Rightarrow$ Q),	hyp,	$\{2\}, \{\}$	H
3	! B(Mary, P $\Rightarrow$ R),	hyp,	$\{3\}, \{\}$	H
4	! B(Mary, Q) $\Rightarrow$ S,	hyp,	$\{4\}, \{\}$	H
5	! B(Mary, R) $\Rightarrow$ $\neg$S,	hyp,	$\{5\}, \{\}$	H
				open Mary
11	‽B(Mary, Q),	sim,	$\{1, 2\}, \{11\}$	BI 9
12	‽B(Mary, R),	sim,	$\{1, 3\}, \{12\}$	BI 10
13	‽S,	der,	$\{1, 2, 4\}, \{11\}$	$\Rightarrow$E 4,11
14	‽$\neg$S,	der,	$\{1, 3, 5\}, \{12\}$	$\Rightarrow$E 5,12
15	‽S $\wedge$ $\neg$S,	der,	$\{1, 2, 3, 4, 5\}, \{11, 12\}$	$\wedge$I 13,14

	Mary (Cassie), Mary, $6/\{1\}\{\}, 7/\{2\}\{\}, 8/\{3\}\{\}$			
6	! P,	hyp,	$\{6\}, \{\}$	SH 1
7	! P $\Rightarrow$ Q,	hyp,	$\{7\}, \{\}$	SH 2
8	! P $\Rightarrow$ R,	hyp,	$\{8\}, \{\}$	SH 3
9	! Q,	der,	$\{6, 7\}, \{\}$	$\Rightarrow$E 6,7
10	! R,	der,	$\{6, 8\}, \{\}$	$\Rightarrow$E 6,8

$\sigma = \langle\!\langle \mathsf{I}, \{1, 2, 3, 4, 5\}, \{\}, \top \rangle\!\rangle$
$P_1(\sigma) = \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{Q}), ?, \mathtt{sim}, \{1, 2\}, \{11\}\rangle, \langle !\mathsf{B}(\mathsf{Mary}, \mathsf{R}), ?, \mathtt{sim}, \{1, 3\}, \{12\}\rangle, \ldots\}$
$\mathcal{A}(\sigma) =_{def} \{P_1(\sigma)\}$

Figure 5.5: Simulation with multiple extensions

1. *The belief state* $\langle\!\langle a, H \cup sent(E), \{\}, \sigma_p \rangle\!\rangle$ *is consistent, and*

2. $\bigcup_{s \in E} \alpha(s) \subseteq sent(E)$*, i.e., E is closed ($\alpha(s)$ selects the assumption origin set of s).*

By virtue of the closure condition, not only the individual sentences of an extension but also all the assumptions they are based on must be consistent with the belief state σ.

Reconsidering the previous example in light of this new definition leads to the believabilities shown in Figure 5.6. Now they are computed according to the multiple extensions $\mathcal{E}_1(\sigma)$,which makes the contradiction in step 15 unbelievable and everything else based on the two simulation assumptions only possible. Intuitively, two sentences that are in different extensions cannot be believed by Cassie "in the same breath". An extension can be viewed as defining a frame of mind where two sentences in different frames of mind might be believable individually, but their conjunction is only believable if they are in one frame together.

5.4.7 Unrestricted Extensions

Throughout, the main premise of our simulative reasoning approach has been that Cassie simulates

	Cassie ($\top$), I			
1	! B(Mary, P),	hyp,	$\{1\}, \{\}$	H
2	! B(Mary, P $\Rightarrow$ Q),	hyp,	$\{2\}, \{\}$	H
3	! B(Mary, P $\Rightarrow$ R),	hyp,	$\{3\}, \{\}$	H
4	! B(Mary, Q) $\Rightarrow$ S,	hyp,	$\{4\}, \{\}$	H
5	! B(Mary, R) $\Rightarrow$ $\neg$S,	hyp,	$\{5\}, \{\}$	H
				open Mary
11	≈!B(Mary, Q),	sim,	$\{1, 2\}, \{11\}$	BI 9
12	≈!B(Mary, R),	sim,	$\{1, 3\}, \{12\}$	BI 10
13	≈!S,	der,	$\{1, 2, 4\}, \{11\}$	$\Rightarrow$E 4,11
14	≈!$\neg$S,	der,	$\{1, 3, 5\}, \{12\}$	$\Rightarrow$E 5,12
15	⊛S $\wedge$ $\neg$S,	der,	$\{1, 2, 3, 4, 5\}, \{11, 12\}$	$\wedge$I 13,14

	Mary (Cassie), Mary, 6/{1}{}, 7/{2}{}, 8/{3}{}			
6	! P,	hyp,	$\{6\}, \{\}$	SH 1
7	! P $\Rightarrow$ Q,	hyp,	$\{7\}, \{\}$	SH 2
8	! P $\Rightarrow$ R,	hyp,	$\{8\}, \{\}$	SH 3
9	! Q,	der,	$\{6, 7\}, \{\}$	$\Rightarrow$E 6,7
10	! R,	der,	$\{6, 8\}, \{\}$	$\Rightarrow$E 6,8

$$\sigma = \langle\!\langle \mathsf{I}, \{1,2,3,4,5\}, \{\}, \top \rangle\!\rangle$$
$$P_1(\sigma) = \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{Q}), ?, \mathtt{sim}, \{1,2\}, \{11\}\rangle, \langle !\mathsf{B}(\mathsf{Mary}, \mathsf{R}), ?, \mathtt{sim}, \{1,3\}, \{12\}\rangle, \ldots\}$$
$$\mathcal{E}_1(\sigma) = \{\{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{Q}), ?, \mathtt{sim}, \{1,2\}, \{11\}\rangle, \ldots\}, \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{R}), ?, \mathtt{sim}, \{1,3\}, \{12\}\rangle, \ldots\}\}$$
$$\mathcal{A}(\sigma) =_{def} \mathcal{E}_1(\sigma)$$

Figure 5.6: Simulation with believabilities according to multiple extensions

the reasoning of other agents by attributing her own reasoning skills to the agents simulated by her just as she would use these skills herself. The same should of course also apply in cases of nested simulations. However, as the example in Figure 5.7 shows, this is not yet the case, since our current definition of extensions only goes one level deep.

The believabilities in the Cassie and the Mary context were computed from the one-level extensions determined by the belief states σ and σ_M that describe each context individually. While the sentence !B(Sally, Q) in step 10 is unbelievable in the Mary context, it becomes plausibly believable once it gets introduced as a simulation result into the top-level Cassie context. This is undesirable; hence, we have to change the definition of extensions to take simulations in nested contexts into account.

There are actually two scenarios in which the extensions of one belief state might influence the extensions of another:

1. The extensions of a simulation (or child) belief state influence the extensions of the simulator (or parent) belief state. This is the case demonstrated by the example in Figure 5.7.

	Cassie (⊤), I			
1	! B(Mary, B(Sally, P)),	hyp,	{1}, {}	H
2	! B(Mary, B(Sally, P ⇒ Q)),	hyp,	{2}, {}	H
3	! B(Mary, ¬B(Sally, Q)),	hyp,	{3}, {}	H
				open Mary
11	!B(Mary, B(Sally, Q)),	sim,	{1, 2}, {11}	BI 10

	Mary (Cassie), Mary, 4/{1}{}, 5/{2}{}, 6/{3}{}			
4	! B(Sally, P),	hyp,	{4}, {}	SH 1
5	! B(Sally, P ⇒ Q),	hyp,	{5}, {}	SH 2
6	! ¬B(Sally, Q),	hyp,	{6}, {}	SH 3
				open MarySally
10	!B(Sally, Q),	sim,	{4, 5}, {10}	BI 9

	MarySally (Mary), Sally, 7/{4}{}, 8/{5}{}			
7	! P,	hyp,	{7}, {}	SH 4
8	! P ⇒ Q,	hyp,	{8}, {}	SH 5
9	! Q,	der,	{7, 8}, {}	⇒E 7,8

$\sigma = \langle\!\langle \mathsf{I}, \{1,2,3\}, \{\}, \top \rangle\!\rangle$
$P_1(\sigma) = \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{1,2\}, \{11\}\rangle, \ldots\}$
$\mathcal{E}_1(\sigma) = \{P_1(\sigma)\}$
$\mathcal{A}(\sigma) =_{def} \mathcal{E}_1(\sigma)$

$\sigma_M = \langle\!\langle \mathsf{Mary}, \{4,5,6\}, \{\}, \sigma \rangle\!\rangle$
$P_1(\sigma_M) = \{\ldots\}$
$\mathcal{E}_1(\sigma_M) = \{P_1(\sigma_M)\}$
$\mathcal{A}(\sigma_M) =_{def} \mathcal{E}_1(\sigma_M)$

Figure 5.7: Nested simulations need multi-level extensions

2. The extensions of a simulator belief state influence the extensions of one of its children. This case can arise if the simulation of one agent leads to the derivation of belief sentences that get used in the simulation of another agent as a priori simulation assumptions. We already had an example of such a case in Figure 5.4, where the sentence in step 11 in the Sally context should be unbelievable, because it is based on an unbelievable sentence in the Cassie context. This problem is partly taken care of by the fact that once a simulation result based on that sentence gets introduced into the parent context, it will automatically become unbelievable, because its origin set contains an unbelievable sentence. However, from a cognitive modeling perspective, nothing should be believable in a simulation context that is not based on something believable in the simulator context.

Definition 5.4.13. *Let $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$, and let the simulation hypotheses available for the direct simulation of some agent b be given by the following sets:*

$$
\begin{aligned}
H^\star_{b,\sigma} &= \{!p \mid \sigma \vdash \langle !\mathsf{B}(b,p), a/?, \tau, \omega, \{\} \rangle\} \\
A^\star_{b,\sigma} &= \{!p \mid \sigma \vdash \langle !\mathsf{B}(b,p), a/?, \tau, \omega, \alpha \rangle, \alpha \neq \emptyset\}
\end{aligned}
$$

Then $\sigma_b = \langle\!\langle b, H^\star_{b,\sigma}, A^\star_{b,\sigma}, \sigma \rangle\!\rangle$ is the **simulation belief state** *for agent b in σ.*

The simulation belief state for some agent b specifies the set of hypotheses and assumptions that can be introduced into the simulation context for that agent via the rules SH and SH$'$.

As before, we define unrestricted or multi-level extensions in two steps. Unrestricted prima facie extensions are the same as one-level prima facie extensions; i.e., they only filter out the simulation assumptions that obviously cannot be part of any extension of a belief state:

Definition 5.4.14. *Let $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$. Then the* **prima facie extension** *$P(\sigma) = P_1(\sigma)$.*

Unrestricted extensions partition the simulation assumptions in the deductive closure of a belief state into subsets of "reasonable" assumptions. What is reasonable is defined in terms of derivability of certain sentences in simulation contexts at arbitrary depths. Even if a belief state σ contains only a finite set of hypotheses, there is no upper bound to the level of nesting of simulation contexts used to derive the elements of its deductive closure, since, for example, hypothetical reasoning can introduce arbitrarily nested belief sentences. For this reason, we define unrestricted extensions iteratively, thus considering deeper and deeper nested simulations with every iteration:

Definition 5.4.15. *Let $\sigma = \langle\!\langle a, H, A, \sigma_p \rangle\!\rangle$. Its proper* **extensions** *$\mathcal{E}(\sigma)$ are defined incrementally with $\mathcal{E}_i(\sigma)$ referring to their state at iteration $i, i \geq 1$:*

$\mathcal{E}_i(\top) = \{\{\}\}$ for all $i \geq 1$.

$\mathcal{E}_1(\sigma) = \{E \subseteq P(\sigma)\}$ where each E is a maximal subset of $P(\sigma)$ such that:

1. *The belief state $\langle\!\langle a, H \cup sent(E), \{\}, \sigma_p \rangle\!\rangle$ is consistent, and*
2. *$\bigcup_{s \in E} \alpha(s) \subseteq sent(E)$, i.e., E is closed ($\alpha(s)$ selects the assumption origin set of s).*

$\mathcal{E}_{i+1}(\sigma) = \{E \subseteq P(\sigma)\}$ where each E is a maximal subset of $P(\sigma)$ such that:

1. *The belief state $\langle\!\langle a, H \cup sent(E), \{\}, \sigma_p \rangle\!\rangle$ is consistent, and*
2. *$\bigcup_{s \in E} \alpha(s) \subseteq sent(E)$, and*
3. *there exists an extension $E_p \in \mathcal{E}_i(\sigma_p)$ such that*
   $\bigcup_{s \in E} \alpha(s) \cap A \subseteq \{!q \mid !\mathsf{B}(b,q) \in sent(E_p)\}$, and
4. *for each $!\mathsf{B}(b,p) \in sent(E)$ there exists an extension $E_b \in \mathcal{E}_i(\sigma_b)$ such that*
   $\{!q \mid \langle !\mathsf{B}(b,q), a/?, \tau, \omega, \alpha \rangle \in \sigma^\star, \alpha \in sent(E)\}) \subseteq \{!q \mid \langle !q, b/?, \tau, \omega, \alpha \rangle \in \sigma^\star_b, \alpha \in sent(E_b)\})$

$\mathcal{E}(\sigma) = \mathcal{E}_l(\sigma)$ for the smallest $l \geq 1$ for which $\mathcal{E}_l(\sigma) = \mathcal{E}_{l+k}(\sigma)$ for all $k \geq 1$.

The central parts of the above definition that make unrestricted extensions different from one-level extensions are the third and fourth condition in the induction step. Condition three can be paraphrased as follows: A set of sentences can only be an extension of a belief state σ if the combination of a priori simulation assumptions on which the sentences of the extension are based have associated belief sentences $!\mathsf{B}(b, q)$ that are believable simultaneously in at least one extension of the parent context σ_p. For example, if Cassie believes $!\mathsf{B}(\mathsf{Mary}, \mathsf{P})$ and $!\mathsf{B}(\mathsf{Mary}, \mathsf{Q})$, but these two sentences are in different extensions, which means that she can never believe them simultaneously, then no simulation result in the Mary context which is based on both $!\mathsf{P}$ and $!\mathsf{Q}$ should ever be believable there. Said differently, Cassie should never be able to believe $!\mathsf{P}$ and $!\mathsf{Q}$ simultaneously during the simulation of Mary, since she can never simultaneously believe the reasons for these simulation hypotheses. Thus, condition three ensures that the extensions of the simulation context are properly constrained by the extensions of the simulator or parent context.

Condition four expresses the following: A set of sentences can only be an extension of a belief state σ if for every set of sentences $!\mathsf{B}(b, p)$ believable in σ extended by the extension, there is at least one extension in the associated simulation belief state σ_b in which all the object propositions of those belief sentences are believable simultaneously. For example, if $!\mathsf{P}$ and $!\mathsf{Q}$ are in different extensions in the Mary context, then $!\mathsf{B}(\mathsf{Mary}, \mathsf{P})$ and $!\mathsf{B}(\mathsf{Mary}, \mathsf{Q})$ should wind up in different extensions in the parent context of Mary. Thus, condition four goes into the opposite direction from condition three, since it ensures that the extensions in the simulator or parent context are properly constrained by the extensions of the various simulation or child contexts.

Taken together, conditions three and four ensure that consistency constraints for the definition of extensions flow properly up and down across simulation context boundaries.

Equipped with the concept of unrestricted extensions we can recompute the believabilities from the previous example as shown in Figure 5.8. This time, the sentence $!\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q}))$ cannot be in any extension of σ, since its object proposition $!\mathsf{B}(\mathsf{Sally}, \mathsf{Q})$ cannot be in any extension of σ_M.

Figure 5.9 shows how the extension mechanism resolves the problem of mutually contradicting simulations. Here we have the case that due to two admittedly nonsensical rules in steps 5 and 6 Cassie believes that whenever Mary believes that Sally believes some proposition then Lucy doesn't believe that and vice versa. While the simulation results in steps 13 and 20 are both elements of the prima facie extension of σ, they cannot be in the same extensions, since the object propositions of the belief sentences in steps 13 and 21 can never be simultaneously in any extension of belief state σ_M (and, symmetrically, the same holds for sentences 20 and 22 and belief state σ_L). Note, that despite the different numbers in the sentence sets the belief states σ_M and σ_L are almost identical, since the numbers are only names for the plain sentences on the referenced lines. For example, 7 and 14 are two different names for the sentence $!\mathsf{B}(\mathsf{Sally}, \mathsf{P})$. The only real difference between σ_M and σ_L is their reasoning agent.

5.4.8 The Parsimonious Shadowing Anomaly

Figure 5.10 shows a situation which, at least on first sight, seems to demonstrate a misfeature of the current shadowing approach. As shown, the shadowing that takes place after sentence 11 gets added is parsimonious, since only the directly contradicting sentence 8 gets shadowed, but not 9 and 10. This is completely in line with the definition of prima facie extensions and extensions: Only if we add 8 as a hypothesis to the belief state defined by 1, 2 and 11 can we derive a real

	Cassie ($\top$), I			
1	$!\,\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{P}))$,	hyp,	$\{1\}, \{\}$	H
2	$!\,\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{P} \Rightarrow \mathsf{Q}))$,	hyp,	$\{2\}, \{\}$	H
3	$!\,\mathsf{B}(\mathsf{Mary}, \neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q}))$,	hyp,	$\{3\}, \{\}$	H
				open Mary
11	⊛$\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q}))$,	sim,	$\{1,2\}, \{11\}$	BI 10

	Mary (Cassie), Mary, $4/\{1\}\{\}, 5/\{2\}\{\}, 6/\{3\}\{\}$			
4	$!\,\mathsf{B}(\mathsf{Sally}, \mathsf{P})$,	hyp,	$\{4\}, \{\}$	SH 1
5	$!\,\mathsf{B}(\mathsf{Sally}, \mathsf{P} \Rightarrow \mathsf{Q})$,	hyp,	$\{5\}, \{\}$	SH 2
6	$!\,\neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q})$,	hyp,	$\{6\}, \{\}$	SH 3
				open MarySally
10	⊛$\mathsf{B}(\mathsf{Sally}, \mathsf{Q})$,	sim,	$\{4,5\}, \{10\}$	BI 9

	MarySally (Mary), Sally, $7/\{4\}\{\}, 8/\{5\}\{\}$			
7	$!\,\mathsf{P}$,	hyp,	$\{7\}, \{\}$	SH 4
8	$!\,\mathsf{P} \Rightarrow \mathsf{Q}$,	hyp,	$\{8\}, \{\}$	SH 5
9	$!\,\mathsf{Q}$,	der,	$\{7,8\}, \{\}$	$\Rightarrow$E 7,8

$\sigma = \langle\!\langle \mathsf{I}, \{1,2,3\}, \{\}, \sigma_p \rangle\!\rangle$
$P(\sigma) = \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{1,2\}, \{11\}\rangle, \ldots\}$
$\mathcal{E}(\sigma) = \{\ldots\}$
$\mathcal{A}(\sigma) =_{def} \mathcal{E}(\sigma)$

$\sigma_M = \langle\!\langle \mathsf{Mary}, \{4,5,6\}, \{\}, \sigma_p \rangle\!\rangle$
$P(\sigma_M) = \{\ldots\}$
$\mathcal{E}(\sigma_M) = \{P(\sigma_M)\}$
$\mathcal{A}(\sigma_M) =_{def} \mathcal{E}(\sigma_M)$

Figure 5.8: Nested simulation with proper believabilities

contradiction. The contradictions derivable by adding either 9 or 10 will only be apparent, that is, they will involve an assumption.

This seems strange for two reasons: (1) Both 9 and 10 were derived with help of the object proposition of 8, which is now shadowed, and (2) it suggests that Mary could plausibly believe a conjunction without believing one of its conjuncts. Both of these observations originate from information only available to us but outside of the logic:

(1) That 9 and 10 were derived with help of the object proposition of 8 is visible to us who did the derivation, and it is recorded in the external explanations of individual inference steps. The origin sets, however, only record what propositions needed to be believed in order to derive the various results, hence, the hypothesis origin sets of 8, 9 and 10 are identical. Thus, there is no way

	Cassie (⊤), I			
1	! B(Mary, B(Sally, P)),	hyp,	{1}, {}	H
2	! B(Mary, B(Sally, P ⇒ Q)),	hyp,	{2}, {}	H
3	! B(Lucy, B(Sally, P)),	hyp,	{3}, {}	H
4	! B(Lucy, B(Sally, P ⇒ Q)),	hyp,	{4}, {}	H
5	! B(Mary, B(Sally, Q)) ⇒ B(Lucy, ¬B(Sally, Q)),	hyp,	{5}, {}	H
6	! B(Lucy, B(Sally, Q)) ⇒ B(Mary, ¬B(Sally, Q)),	hyp,	{6}, {}	H
				open Mary
13	!?B(Mary, B(Sally, Q)),	sim,	{1, 2}, {13}	BI 12
				open Lucy
20	!?B(Lucy, B(Sally, Q)),	sim,	{3, 4}, {20}	BI 19
21	!?B(Mary, ¬B(Sally, Q)),	der,	{3, 4, 6}, {20}	⇒E 6,20
22	!?B(Lucy, ¬B(Sally, Q)),	der,	{1, 2, 5}, {13}	⇒E 5,13

	Mary (Cassie), Mary, 7/{1}{}, 8/{2}{}, 23/{3, 4, 6}{20}			
7	! B(Sally, P),	hyp,	{7}, {}	SH 1
8	! B(Sally, P ⇒ Q),	hyp,	{8}, {}	SH 2
				open...
12	!?B(Sally, Q),	sim,	{7, 8}, {12}	BI 11
23	!?¬B(Sally, Q),	sim,	{}, {23}	SH′ 21

	MarySally (Mary), Sally, 9/{7}{}, 10/{8}{}		
9	! P,	hyp,	{9}, {}
10	! P ⇒ Q,	hyp,	{10}, {}
11	! Q,	der,	{9, 10}, {}

	Lucy (Cassie), Lucy, 14/{3}{}, 15/{4}{}, 24/{1, 2, 5}{13}			
14	! B(Sally, P),	hyp,	{14}, {}	SH 3
15	! B(Sally, P ⇒ Q),	hyp,	{15}, {}	SH 4
				open...
19	!?B(Sally, Q),	sim,	{14, 15}, {19}	BI 18
24	!?¬B(Sally, Q),	sim,	{}, {24}	SH′ 22

	LucySally (Lucy), Sally, 16/{14}{}, 17/{15}{}		
16	! P,	hyp,	{16}, {}
17	! P ⇒ Q,	hyp,	{17}, {}
18	! Q,	der,	{16, 17}, {}

$\sigma = \langle\!\langle \mathsf{I}, \{1,2,3,4,5,6\}, \{\}, \top \rangle\!\rangle$
$P(\sigma) = \{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{1,2\}, \{13\}\rangle, \langle !\mathsf{B}(\mathsf{Lucy}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{3,4\}, \{20\}\rangle, \ldots\}$
$\mathcal{E}(\sigma) = \{\{\langle !\mathsf{B}(\mathsf{Mary}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{1,2\}, \{13\}\rangle, \ldots\}, \{\langle !\mathsf{B}(\mathsf{Lucy}, \mathsf{B}(\mathsf{Sally}, \mathsf{Q})), ?, \mathtt{sim}, \{3,4\}, \{20\}\rangle, \ldots\}\}$
$\mathcal{A}(\sigma) =_{def} \mathcal{E}(\sigma)$

$\sigma_M = \langle\!\langle \mathsf{Mary}, \{7,8\}, \{23\}, \sigma \rangle\!\rangle$
$P(\sigma_M) = \{\langle !\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{7,8\}, \{12\}\rangle, \langle !\neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{\}, \{23\}\rangle, \ldots\}$
$\mathcal{E}(\sigma_M) = \{\{\langle !\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{7,8\}, \{12\}\rangle, \ldots\}, \{\langle !\neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{\}, \{23\}\rangle, \ldots\}\}$
$\mathcal{A}(\sigma_M) =_{def} \mathcal{E}(\sigma_M)$

$\sigma_L = \langle\!\langle \mathsf{Lucy}, \{14,15\}, \{24\}, \sigma \rangle\!\rangle$
$P(\sigma_L) = \{\langle !\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{14,15\}, \{19\}\rangle, \langle !\neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{\}, \{24\}\rangle, \ldots\}$
$\mathcal{E}(\sigma_L) = \{\{\langle !\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{14,15\}, \{19\}\rangle, \ldots\}, \{\langle !\neg\mathsf{B}(\mathsf{Sally}, \mathsf{Q}), ?, \mathtt{sim}, \{\}, \{24\}\rangle, \ldots\}\}$
$\mathcal{A}(\sigma_L) =_{def} \mathcal{E}(\sigma_L)$

Figure 5.9: Mutually contradicting simulations

	Cassie (⊤), I			
1	! B(Mary, P),	hyp,	{1}, {}	H
2	! B(Mary, P ⇒ Q),	hyp,	{2}, {}	H
				open Mary
8	! B(Mary, Q),	sim,	{1, 2}, {8}	BI 5
9	! B(Mary, Q ∧ Q),	sim,	{1, 2}, {9}	BI 6
10	! B(Mary, Q ∨ R),	sim,	{1, 2}, {10}	BI 7
11	! ¬B(Mary, Q),	hyp,	{11}, {}	H

	Mary (Cassie), Mary, 3/{1}{}, 4/{2}{}			
3	! P,	hyp,	{3}, {}	SH 1
4	! P ⇒ Q,	hyp,	{4}, {}	SH 2
5	! Q,	der,	{3, 4}, {}	⇒E 3,4
6	! Q ∧ Q,	der,	{3, 4}, {}	∧I 5
7	! Q ∨ R,	der,	{3, 4}, {}	∨I 5

Figure 5.10: Parsimonious shadowing

for the logic to tell what intermediate results were necessary to derive a particular sentence, or, differently put, as far as the logic is concerned, there is no causal connection between 8 and 9.

(2) To realize that it is strange to believe 9 but not 8 one has to know something about the "mutual closeness" of these sentences, i.e., that they are only one inference step apart. Again, this is a kind of meta-information available to us but outside of the logic.

One solution to the problems above would be to additionally record the explanations of individual inference steps as part of the support of a sentence instead of just keeping them as external annotations. In truth-maintenance-system terminology, we would add a JTMS to the ATMS. This would mean that with every sentence we would also record the set of proofs that derived it and make the computations of extensions dependent on this new information. It would greatly complicate the theory, but it might be worthwhile to pursue this direction in future extensions of SIMBA.

Another approach would be to try to change the definitions of prima facie extensions and extensions such that sentences 9 and 10 could not be consistently added any more to the belief state defined by 1, 2, and 11. For example, one could strengthen the consistency requirement to only add simulation assumptions to the prima facie extension that would keep the belief state strongly consistent (i.e., they would not even lead to apparent contradictions partially based on assumptions). This would automatically shadow all sentences whose object propositions imply the object proposition of the directly shadowed sentence; in our example, it would shadow 9 though not 10. A problem is that strong consistency could be violated by a completely independent set of sentences that characterize the incomplete beliefs of another agent, in which case it would not be a meaningful criterion anymore. Yet one could call relevance logic to the rescue, since it would allow one to reliably check whether the derived contradiction is based on the added sentence in question or not. However, one can still construct counterexamples where the derived contradiction does depend on the added sentence, but does not have anything to do with the directly shadowed sentence. This argument does of course not prove that no such solution exists; it just shows that the author is not aware of a feasible approach along these lines.

	Cassie ($\top$), I			
1	$!\,\mathsf{B}(\mathsf{Oscar}, \forall \mathsf{c}_1, \mathsf{c}_2, \mathsf{e}\,(\mathsf{Class}(\mathsf{c}_1) \wedge \mathsf{Class}(\mathsf{c}_2) \wedge$			
	$\wedge \mathsf{Equiv}(\mathsf{c}_1, \mathsf{c}_2) \wedge \mathsf{Member}(\mathsf{e}, \mathsf{c}_2)) \Rightarrow \mathsf{Member}(\mathsf{e}, \mathsf{c}_1)),$	`hyp`,	$\{1\}, \{\}$	H
2	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Class}(\mathsf{P})),$	`hyp`,	$\{2\}, \{\}$	H
3	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Class}(\mathsf{NP})),$	`hyp`,	$\{3\}, \{\}$	H
4	$!\,\mathsf{B}(\mathsf{Oscar}, \forall \mathsf{p}\, \mathsf{Member}(\mathsf{p}, \mathsf{P}) \Rightarrow \mathsf{PolynomialTime}(\mathsf{p})),$	`hyp`,	$\{4\}, \{\}$	H
5	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Member}(\mathsf{SAT}, \mathsf{NP})),$	`hyp`,	$\{5\}, \{\}$	H
				open...
18	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}) \Rightarrow \mathsf{PolynomialTime}(\mathsf{SAT})),$	`sim`,	$\{1, 2, 3, 4, 5\}, \{18\}$	BI 17
	...			Exam
99	$!\,\neg\mathsf{B}(\mathsf{Oscar}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}) \Rightarrow \mathsf{PolynomialTime}(\mathsf{SAT})),$	`hyp`,	$\{99\}, \{\}$	H

Figure 5.11: Shadowing can lead to information loss

5.5 Belief vs. Belief Entailment

Using the newly defined concept of believability, we can look again at the initial motivating example from Figure 5.1. Cassie's top-level reasoning context is shown again in Figure 5.11, but now with proper believabilities added to individual top-level sentences. As expected, the problem with the contradiction between the simulation result in step 18 and the directly acquired belief hypothesis in step 99 gets solved by shadowing the simulation result.

What is disconcerting, however, is that the shadowing process not only shadowed the contradicting conclusion, but with it the very fact that it was possible to derive such a contradiction in the first place. In the above situation, a human teacher would have observed not only that Oscar did not believe the sentence in question, but also that Oscar failed to perform up to the teacher's expectations. Such differences between expectations and observed facts carry a great deal of useful information. In an actual teaching situation, such an observation probably would have led to corrective pedagogical action by the teacher. Cassie, on the other hand, forgot everything about Oscar's simulation result the very instant she started to believe sentence $\mathsf{99}$.

Nutter [1983] critizices nonmonotonic logics in general by saying that they cannot represent the situation that "there is reason to suppose C, but not C". She observes that the most interesting thing about the fact that Opus the penguin is a bird who does not fly is not that he simply fails to fly, but that the default rule telling us that birds usually fly does not apply to penguins. According to that rule, there is reason to suppose that Opus flies, but nevertheless he doesn't. In a standard default logic, Nutter argues, "there is reason to suppose that Opus flies" becomes identical to "Opus flies"; thus, the failing of a default rule cannot be expressed explicitly, since it would cause an outright contradiction.

This is exactly the kind of shortcoming exhibited by the current version of **SL**. The simulation of some agent's reasoning gives Cassie reason to believe that the agent actually believes the simulation result. This information should not be lost even if the simulation result gets shadowed by evidence to the contrary.

5.5.1 Belief Entailment

To enable Cassie to reason about a situation in which an agent does not hold a certain belief even though the agent's other beliefs entail it, we introduce a new propositional function called *belief entailment*, or BE. Belief entailment is syntactically very similar to the belief function used so far.

The main difference between the two is their semantics. By saying that $\mathsf{BE}(\mathsf{Mary}, \mathsf{Nice}(\mathsf{John}))$ we mean that the propositions believed by Mary entail that John is nice, or, said differently, according to what we know about Mary's beliefs she ought to believe that John is nice, or, to stick with Nutter's terminology, Mary has reason to believe that John is nice. The difference between belief and belief entailment is best expressed from Cassie's point of view:

- $!\mathsf{B}(b, q)$ means "I believe that b believes q".
- $!\mathsf{BE}(b, q)$ means "if I were b, I would believe q".

Formally, we define the denotation of BE as the domain function $\mathbf{f_{BE}}$. The specification of an admissible interpretation function *int* has to be extended such that $int(\mathsf{BE}) = \mathbf{f_{BE}}$. The definition of $\mathbf{f_{BE}}$ is similar to Definition 4.3.4, which specified the denotation of the belief function:

Definition 5.5.1. *Let $\boldsymbol{a} \in \mathcal{D}_{a_{np}}$ and $\boldsymbol{p} \in \mathcal{D}_p$. The* **belief entailment function**, *$\mathbf{f_{BE}}$, is given as:*

$\mathbf{f_{BE}}$ *: $\mathcal{D}_{a_{np}} \times \mathcal{D}_p \to \mathcal{D}_{s_p}$. The value of $\mathbf{f_{BE}}(\boldsymbol{a}, \boldsymbol{p})$ is the proposition that the propositions believed by $\boldsymbol{a}$ entail $\boldsymbol{p}$.*

The definition of $\models_a !\mathsf{BE}(b, t)$ is identical to the one for B; that is, $\models_a !\mathsf{BE}(b, t)$ iff $\models_a !\mathsf{B}(b, t)$, since the semantic account of $\mathbf{SL}_0$ does not consider different believabilities. As will become clear below, in the monotonic logic $\mathbf{SL}_0$, the functions B and BE would have been almost identical with the exception of the definitions of their associated domain functions $\mathbf{f_B}$ and $\mathbf{f_{BE}}$.

Just as everything said so far about BE was very similar to what was said about B, so is the inference rule used to introduce belief entailment sentences which is defined below. The main difference is that belief entailments can be introduced as plain derived conclusions or as simulation assumptions depending on whether the simulation of the agent's reasoning involved a nested simulation or not. If no nested simulation was involved, i.e., the only assumptions in the origin set of $!q$ are a priori assumptions that were introduced with the rule SH$'$, then the introduced result will be a plain derived sentence. If a nested simulation was involved then the introduced result will become a simulation assumption. This mirrors exactly the way Cassie reasons at her top level but shifted by one level. It exemplifies precisely the semantics we defined above.

Belief Entailment Introduction (BEI)

$$
\begin{array}{r|l||l}
\hline
 & C\ (\ldots),\ a & \\
\hline
 & \ldots & \\
l+1 & !\mathsf{BE}(b,q),\ \ \mathtt{der},\ \ \bigcup\limits_{!p_i \in \omega\cup\alpha} \omega_i,\ \ \bigcup\limits_{!p_i \in \omega\cup\alpha} \alpha_i & \mathsf{BEI}\ \ m \\
 & \ldots & \\
l+1' & !\mathsf{BE}(b,q),\ \ \mathtt{sim},\ \ \bigcup\limits_{!p_i \in \omega\cup\alpha} \omega_i,\ \{!\mathsf{BE}(b,q)\} \cup \bigcup\limits_{!p_i \in \omega\cup\alpha} \alpha_i & \mathsf{BEI}'\ \ m \\
\hline
\end{array}
$$

$$
\begin{array}{r||l||}
\hline
 & D\ (C),\ b, \\
 & !p_1/\omega_1\alpha_1, \ldots, !p_n/\omega_n\alpha_n \\
\hline
 & \ldots \\
m & !q, \qquad \tau,\ \ \omega \subseteq shyps(D,m),\ \alpha \subseteq shyps'(D,m) \\
 & \ldots \\
m' & !q, \qquad \tau,\ \ \omega \subseteq shyps(D,m),\ \alpha \setminus shyps'(D,m) \neq \emptyset \\
\hline
\end{array}
$$

	Cassie (⊤), I			
1	! B(Mary, P),	hyp,	{1}, {}	H
2	! B(Mary, P ⇒ Q),	hyp,	{2}, {}	H
3	! B(Mary, B(Sally, P)),	hyp,	{3}, {}	H
4	! B(Mary, B(Sally, P ⇒ Q)),	hyp,	{4}, {}	H
				open Mary
15	⸸B(Mary, Q),	sim,	{1, 2}, {15}	BI 7
16	! BE(Mary, Q),	der,	{1, 2}, {}	BEI 7
17	⸸B(Mary, B(Sally, Q)),	sim,	{3, 4}, {17}	BI 13
18	⸸BE(Mary, B(Sally, Q)),	sim,	{3, 4}, {18}	BEI 13
19	⸸B(Mary, BE(Sally, Q)),	sim,	{3, 4}, {19}	BI 14
20	! BE(Mary, BE(Sally, Q)),	der,	{3, 4}, {}	BEI 14

	Mary (Cassie), Mary, 5/{1}{},6/{2}{},8/{3}{},9/{4}{}			
5	! P,	hyp,	{5}, {}	SH 1
6	! P ⇒ Q,	hyp,	{6}, {}	SH 2
7	! Q,	der,	{5, 6}, {}	⇒E 5,6
8	! B(Sally, P),	hyp,	{8}, {}	SH 3
9	! B(Sally, P ⇒ Q),	hyp,	{9}, {}	SH 4
				open MarySally
13	⸸B(Sally, Q),	sim,	{8, 9}, {13}	BI 12
14	! BE(Sally, Q),	der,	{8, 9}, {}	BEI 12

	MarySally (Mary), Sally, 10/{8}{},11/{9}{}			
10	! P,	hyp,	{10}, {}	SH 8
11	! P ⇒ Q,	hyp,	{11}, {}	SH 9
12	! Q,	der,	{10, 11}, {}	⇒E 10,11

Figure 5.12: Difference between belief and belief entailment in nested simulations

The example in Figure 5.12 makes clear how belief entailment differs from plain belief in the context of nested simulations. Sentence 15 is a standard simulation result introduced as an assumption. The belief entailment 16, however, was introduced as a derived result not based on any assumptions, since the derivation of its object proposition did not involve any nested simulations in the Mary context. The object proposition of 18, on the other hand, was derived via a nested simulation; hence, in that case, the belief entailment is also introduced as a simulation assumption. Said differently, the belief entailments introduced into the Cassie context are assumptions if and only if their associated object propositions were derived via simulation in the Mary context. This is exactly what we want, since the belief entailments represent *what* Cassie would believe if she really were Mary and did believe exactly what she believes Mary believes, and *how* or with what conviction she would believe it. 19 shows how Cassie can have a belief about a belief entailment

	Cassie (⊤), I			
1	$!\,\mathsf{B}(\mathsf{Oscar}, \forall \mathsf{c}_1, \mathsf{c}_2, \mathsf{e}\,(\mathsf{Class}(\mathsf{c}_1) \wedge \mathsf{Class}(\mathsf{c}_2) \wedge$			
	$\wedge \mathsf{Equiv}(\mathsf{c}_1, \mathsf{c}_2) \wedge \mathsf{Member}(\mathsf{e}, \mathsf{c}_2)) \Rightarrow \mathsf{Member}(\mathsf{e}, \mathsf{c}_1)),$	`hyp`,	$\{1\}, \{\}$	H
2	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Class}(\mathsf{P})),$	`hyp`,	$\{2\}, \{\}$	H
3	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Class}(\mathsf{NP})),$	`hyp`,	$\{3\}, \{\}$	H
4	$!\,\mathsf{B}(\mathsf{Oscar}, \forall \mathsf{p}\, \mathsf{Member}(\mathsf{p}, \mathsf{P}) \Rightarrow \mathsf{PolynomialTime}(\mathsf{p})),$	`hyp`,	$\{4\}, \{\}$	H
5	$!\,\mathsf{B}(\mathsf{Oscar}, \mathsf{Member}(\mathsf{SAT}, \mathsf{NP})),$	`hyp`,	$\{5\}, \{\}$	H
				open...
18	$\circledast\mathsf{B}(\mathsf{Oscar}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}) \Rightarrow \mathsf{PolynomialTime}(\mathsf{SAT})),$	`sim`,	$\{1, 2, 3, 4, 5\}, \{18\}$	BI 17
	...			Exam
99	$!\,\neg\mathsf{B}(\mathsf{Oscar}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}) \Rightarrow \mathsf{PTime}(\mathsf{SAT})),$	`hyp`,	$\{99\}, \{\}$	H
100	$!\,\mathsf{BE}(\mathsf{Oscar}, \mathsf{Equiv}(\mathsf{P}, \mathsf{NP}) \Rightarrow \mathsf{PTime}(\mathsf{SAT})),$	`der`,	$\{1, 2, 3, 4, 5\}, \{\}$	BEI 17

Figure 5.13: Using belief entailment prevents information loss

believed by another agent, and 20 demonstrates how even a nested belief entailment can be a hard conclusion not based on any assumptions.

Now that we have this new belief entailment function at our disposal, we can solve the motivating problem described at the beginning of this section. The result is given in Figure 5.13, which again shows the result of Cassie's simulation of Oscar's reasoning about complexity theory. This time, however, she is able to represent that there is reason to believe that Oscar believes the result in question by her belief in the proposition of sentence 100, but that he nevertheless does not believe it, which is expressed by 99.

5.5.2 Accounting for BE in the Definition of Extensions

Since the rule of belief entailment introduction can generate simulation assumptions that are not accounted for in the current definition of extensions, we have to extend that definition accordingly. All that needs to be changed to account for the new BE function, is to add the following condition to the inductive step of Definition 5.4.15, the rest of the definition remains unchanged:

Definition 5.4.15 contd.

5. *and for each* $!\mathsf{BE}(b, p) \in \mathit{sent}(E)$ *there exists an extension* $E_b \in \mathcal{E}_i(\sigma_b)$ *such that* $\{!q \mid \langle !\mathsf{BE}(b, q), a/?, \tau, \omega, \alpha\rangle \in \sigma^\star, \alpha \in \mathit{sent}(E)\}\} \subseteq \{!q \mid \langle !q, b/?, \tau, \omega, \alpha\rangle \in \sigma_b^\star, \alpha \in \mathit{sent}(E_b)\}\}.$

5.6 Explicitly Representing and Using Believability

Up to now we have kept believability separate from the logic and Cassie's reasoning. A sentence such as $!p$ meant that the agent whose mind contained it (usually taken to be Cassie) believed the proposition denoted by p. Whatever the believability of that sentence was in a particular reasoning context was merely an external annotation based on the computation of extensions, but it did not have any bearing on Cassie's reasoning at all.

But obviously, we want Cassie's reasoning and state of mind to be influenced by the different believabilities of propositions. For example, if her mind contains the sentence $!p$, and the state of her top-level reasoning context renders p unbelievable, then she should not believe p. In a similar vein, if p is unbelievable, then it should never be used as a premise in a rule of inference, since the result will also be unbelievable. .

To achieve this objective we will change our definition of sentences. From now on, a sentence such as $!p$ will be called a *belief candidate*, meaning the agent whose mind contains it considers p to be a candidate for being believed. We will also require that the only way belief candidates can enter the agent's mind is by applying one of the inference rules of $D_{\mathbf{SL}}$. Therefore, every belief candidate is a derived sentence which has at least one support associated with it (hypotheses are trivially derived). Only if a belief candidate $!p$ has at least one associated support that renders p believable in the current belief state of the agent will we say that the agent *believes* the proposition denoted by p.

To characterize this new notion of belief we used the concept of *current belief state* which we have not yet defined. Intuitively, at any point in time Cassie will be in a particular state of reasoning. Any such state will be described by a particular belief state, which is the set of hypotheses and a priori assumptions available in a particular reasoning context. If that reasoning context is the one Cassie is currently focusing on, then its associated belief state is the current belief state. This implies that whether Cassie actually believes a certain proposition or not depends on what state of reasoning she is currently in. Here is a set of definitions that should make these notions more precise:

Definition 5.6.1. *A* **state of reasoning** *for some agent a is a set of reasoning contexts that constitute a legal* **SL** *deduction with highest step number l, and whose top-level context has a as its reasoning agent. At any point in time the state of reasoning that agent a is in is called its* **current state of reasoning**.

Definition 5.6.2. *The reasoning context which is some agent a's focus of attention in its current state of reasoning is called the* **current reasoning context**.

The current reasoning context can usually be thought of as the one in which the premises for an inference rule are considered. The connection between reasoning contexts and belief states is established in the following way:

Definition 5.6.3. *Let C be a reasoning context with reasoning agent a and parent context D in some state of reasoning with highest step number l. Then the* **belief state** *of C is defined as:*

$$bs(C) = \begin{cases} \langle\!\langle a, hyps(C,l), \{\}, \top\rangle\!\rangle & \textit{if } D = \top \\ \langle\!\langle a, hyps(C,l) \cup shyps(C,l), shyps'(C,l), bs(D)\rangle\!\rangle & \textit{otherwise} \end{cases}$$

If C is the current reasoning context of a, then $bs(C)$ is its **current belief state**.

With these concepts available we are ready to define what it means for an agent to believe a particular proposition p:

Definition 5.6.4. *Let C be the current reasoning context of some agent a in a state of reasoning with highest step number l, and let $!p$ be a belief candidate. a* **believes** *p iff $!p \in sent^*(C,l)$ and at least one of its associated supports renders it believable in the current belief state $bs(C)$.*

Therefore, whether the proposition of $!p$ is believed depends on whether $!p$ is one of the previously derived sentences, and whether any of the supports associated with these previous derivations renders the proposition believable in the current belief state. Note, that this notion of belief is now *dynamic*, since the belief state of a reasoning context changes over time when new hypotheses get added to it. This is of course exactly what we want, since, for example, once a particular proposition of a belief candidate has become unbelievable according to the current belief state, it should not be believed anymore.

All this notion of belief requires is that there be at least one extension (or frame of mind) in which the assumptions of the support of a particular belief candidate are contained. This means that Cassie would consider all propositions as being believed with equal conviction. This is of course undesirable, since she should be able to base decisions on whether a particular proposition is believed with certainty or not. In order to allow Cassie to explicitly reason about different believabilities of various propositions, we introduce the special believability functions Cert, Plaus, and Poss. For example, a sentence such as $!\mathsf{Plaus}(p)$ means that the believability of p is at least plausible. The introduction of believability functions is governed by the following inference rules:

Believability Introduction (CI,PsI,PoI)

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p,$	$\tau,$	$\omega, \{\}$	
	$\ldots$			
$l+1$	$!\mathsf{Cert}(p),$	$\mathtt{der},$	$\omega, \{\}$	CI m
$l+1$	$!\mathsf{Plaus}(p),$	$\mathtt{der},$	$\omega, \{\}$	PsI m
$l+1$	$!\mathsf{Poss}(p),$	$\mathtt{der},$	$\omega, \{\}$	PoI m

	$C\ (\ldots), a$			
	$\ldots$			
m	$!p,$	$\tau,$	$\omega, \alpha \neq \emptyset$	
	$\ldots$			
$l+1$	$!\mathsf{Plaus}(p),$	$\mathtt{sim},$	$\omega, \{!\mathsf{Plaus}(p)\} \cup \alpha$	PsI′ m
$l+1$	$!\mathsf{Poss}(p),$	$\mathtt{der},$	ω, α	PoI m

Since stronger believabilities include weaker ones, a proposition of a sentence whose derivation was solely based on hypotheses will be classified as certain, plausible, as well as possible. However, if it has a non-empty assumption origin set, it can only be plausible or possible. The distinction between plausible and possible propositions can of course not be made directly by the inference rule, since that distinction depends on the computation of extensions. For possible propositions, nothing special has to be done, since as long as there is at least one extension which keeps the proposition of the premise $!p$ believable, the associated sentence $!\mathsf{Poss}(p)$ will also remain believable, since it has the same support. Once no such extension exists anymore, $!p$ as well as $!\mathsf{Poss}(p)$ will simultaneously become unbelievable, which is exactly what we want. In a sense $!\mathsf{Poss}(p)$ does not tell Cassie very much, since it can be believed if and only if $!p$ is believable.

In the case of plausible propositions things are slightly more complicated, since a proposition that is plausible at one point can become only possible at a later point, but it will still be believable. Thus $!\mathsf{Plaus}(p)$ sentences have to be made into simulation assumptions, because then the definition of extensions can be used to only make them members of any extension if their source proposition is a member of all extensions. Once that is not the case anymore, e.g., if the source proposition has "degraded" from plausible to possible, $!\mathsf{Plaus}(p)$ will not be part of any extension anymore and, hence, becomes unbelievable. This also means that an assumption of the form $!\mathsf{Plaus}(p)$ will always be either in all extensions of a belief state or in none. Here is how Definition 5.4.15 has to be amended by adding a sixth condition to the inductive step to account for plausibility propositions (the fifth condition was needed for belief entailment). The rest of the definition remains unchanged:

Definition 5.4.15 contd.

6. *and for each* $\langle !\mathsf{Plaus}(p), a/?, \tau, \omega, \alpha \rangle \in E$ *it is the case that* $\alpha \setminus \{\mathsf{Plaus}(p)\} \subseteq sent(E_i)$ *for all* $E_i \in \mathcal{E}_i(\sigma)$.

The example in Figure 5.14 shows various ways in which explicitly represented believabilities can be derived in the course of a simulation.

5.7 Approximating Extensions

Our approach to defining extensions shares an ugly problem with other default logics: The definition is based on the notion of consistency, which in a logic with quantification such as **SL** is an undecidable property. Since we want to use **SL** not just as a tool for theoretical analysis, but as the foundation for the implementation of an actual belief reasoning engine, this is a serious misfeature.

However, since we only want to model the reasoning of an agent (as opposed to do theorem proving), we can choose a weaker condition than consistency that is computable and still useful: Instead of checking whether the sentences of an extension are consistent with the hypotheses of a belief state, which in general is impossible, we only require them to be *not known to be inconsistent*. This is similar to the approach taken by [Martins and Shapiro, 1988].

In our implementation of SIMBA whenever a sentence gets added to a reasoning context and that sentence contradicts an already existing one, we recompute approximations of the extensions of all currently open reasoning contexts according to our iterative definition. Since we only have a finite number of sentences and only have to check for overt inconsistency, we do not have to compute closures or go to arbitrary levels of nesting. And since all sentences record in their support on which hypotheses and assumptions they are based, they will automatically change their believability according to the latest extension approximation. With this approach **SL** becomes a kind of dynamic logic. The quality of the extension approximations can be improved by investing more work in detecting inconsistencies. One way to do this would be to do some limited forward inference whenever a new sentence gets derived, in order to detect contradictions that "lurk right around the corner".

	Cassie (⊤), I			
1	! B(Mary, B(Sally, P)),	hyp,	{1}, {}	H
2	! B(Mary, B(Sally, P ⇒ Q)),	hyp,	{2}, {}	H
3	! B(Mary, ¬B(Sally, Q)),	hyp,	{3}, {}	H
				open Mary
16	B(Mary, B(Sally, P ∨ R)),	sim,	{1}, {16}	BI 12
17	! ¬B(Mary, B(Sally, P ∨ R)),	hyp,	{17}, {}	H
18	BE(Mary, Plaus(B(Sally, P ∨ R))),	sim,	{1}, {18}	BEI 14
19	BE(Mary, Plaus(B(Sally, Q))),	sim,	{1,2}, {19}	BEI 11
20	B(Mary, Plaus(B(Sally, P ∨ R))),	sim,	{1}, {20}	BI 14
21	! Cert(B(Mary, B(Sally, P))),	der,	{1}, {}	CI 1

	Mary (Cassie), Mary, 4/{1}{}, 5/{2}{}, 6/{3}{}			
4	! B(Sally, P),	hyp,	{4}, {}	SH 1
5	! B(Sally, P ⇒ Q),	hyp,	{5}, {}	SH 2
6	! ¬B(Sally, Q),	hyp,	{6}, {}	SH 3
				open MarySally
11	B(Sally, Q),	sim,	{4,5}, {11}	BI 9
12	B(Sally, P ∨ R),	sim,	{4}, {12}	BI 10
13	! Cert(B(Sally, P)),	der,	{4}, {}	CI 4
14	Plaus(B(Sally, P ∨ R),	sim,	{4}, {12,14}	PsI 12
15	Plaus(B(Sally, Q)),	sim,	{4,5}, {11,15}	PsI 11

	MarySally (Mary), Sally, 7/{4}{}, 8/{5}{}			
7	! P,	hyp,	{7}, {}	SH 4
8	! P ⇒ Q,	hyp,	{8}, {}	SH 5
9	! Q,	der,	{7,8}, {}	⇒E 7,8
10	! P ∨ R,	der,	{7}, {}	∨I 7

Figure 5.14: Deriving explicitly represented believabilities during a simulation

Chapter 6

Implementation and Examples

SIMBA is implemented as an extension of SNePS, the Semantic Network Processing System [Shapiro, 1979; Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992]. SIMBA uses the same basic representation mechanism for propositions as SNePS, but it extends the SNePS Inference Package (SNIP) to allow simulative reasoning in the way described in the previous chapters.

SNePS is a knowledge representation language[1] that represents entities of interest such as propositions, actions, rules, etc., by well-formed networks. These networks are created with the help of a language called SNePSUL, the SNePS User Language. In SIMBA we will not create SNePS networks directly to represent propositions. Instead, we will retain the language of the logic **SL** as the basic representation mechanism, and translate **SL** sentences into SNePS networks with the help of a special front-end called SNePSLOG [Shapiro *et al.*, 1981; Matos and Martins, 1989]. With the help of SNePSLOG we can completely hide the representational complexities of the SNePS network language, and concentrate on the reasoning aspects by using a familiar predicate-logic-style language.

SNePSLOG parses input sentences and translates them into SNePS networks. It accepts a language that is very similar to $L_{\mathbf{SL}}$. The main difference is that special symbols such as quantifiers or logical connectives are represented in an ASCII character notation that can easily be entered on a computer terminal. Table 6.1 shows the complete mapping between **SL** symbols and their ASCII equivalents used by SNePSLOG.

SL symbol	SNePSLOG equivalent
$\neg$	`~`
$\wedge$	`and`
$\vee$	`or`
$\Rightarrow$	`=>`
$\forall$	`all`
$\exists$	`exists`

Table 6.1: **SL** symbols and their SNePSLOG equivalents

[1]The name "SNePS" is usually used for both the representation language and the actual computer program that implements it. To distinguish different versions of the language as well as their implementations, a version number is appended. At the time of this writing the most recent version of SNePS is SNePS-2.3.

For example, an **SL** proposition such as

$$\forall \mathsf{x}\,((\mathsf{Man}(\mathsf{x}) \vee \mathsf{Woman}(\mathsf{x})) \Rightarrow \mathsf{Human}(\mathsf{x}))$$

would be entered into SNePSLOG as the ASCII string

```
all(x)((Man(x) or Woman(x)) => Human(x)).
```

The SNePSLOG parser would then analyze this string and translate it into a SNePS network of the form shown in Figure 6.1. Similarly, a belief proposition entered into SNePSLOG as

```
B(Lucy, Nice(John))
```

would be translated into a SNePS network of the form shown in Figure 6.2. We only show these

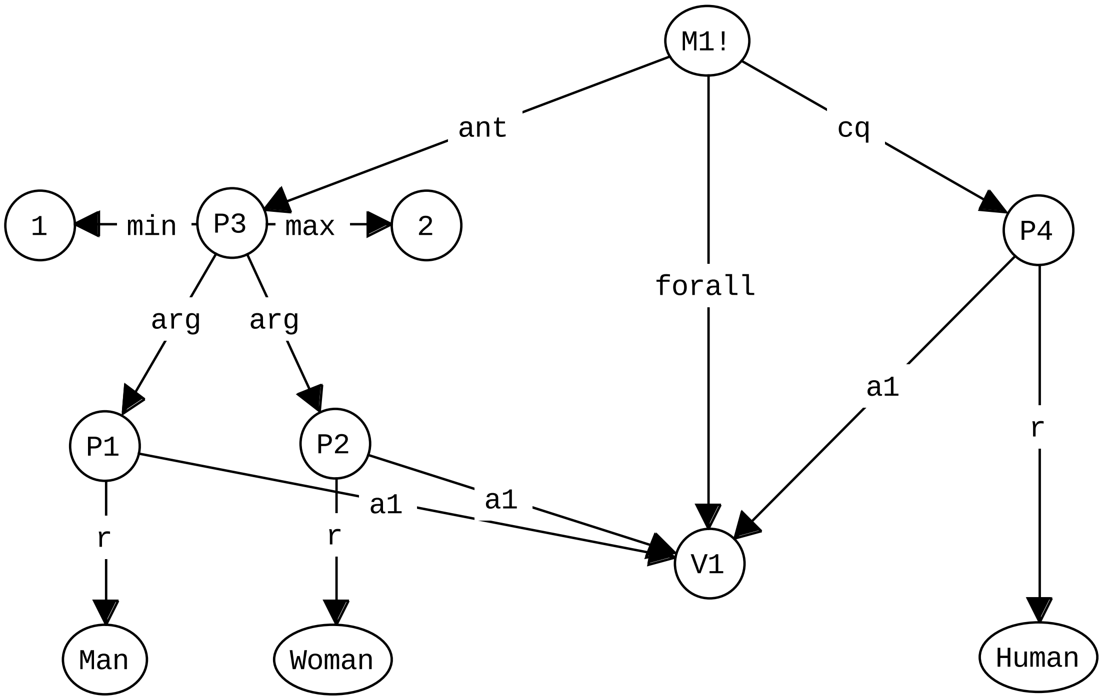


Figure 6.1: Representing `all(x)((Man(x) or Woman(x)) => Human(x))` as a SNePS network

networks for completeness, but it is not necessary to fully understand them in order to understand the presentation of SIMBA that is about to follow. The reader interested in a more detailed definition and analysis of various SNePS networks is referred to the SNePS-related literature cited above. The only aspect of this translation that is relevant in the following is that every proposition is represented by a unique SNePS node, which is the one that *dominates* all nodes of the network generated by the SNePSLOG parser. For example, the node `M1` in Figure 6.1 (it is postfixed with an exclamation mark, since it is believed) is the one that represents the proposition $\forall \mathsf{x}\,((\mathsf{Man}(\mathsf{x}) \vee \mathsf{Woman}(\mathsf{x})) \Rightarrow \mathsf{Human}(\mathsf{x}))$, since it is the root of the generated network from which all other nodes in that network can be reached by only following downward arcs (only those are displayed). This justifies us in talking about SNePS nodes representing propositions without having

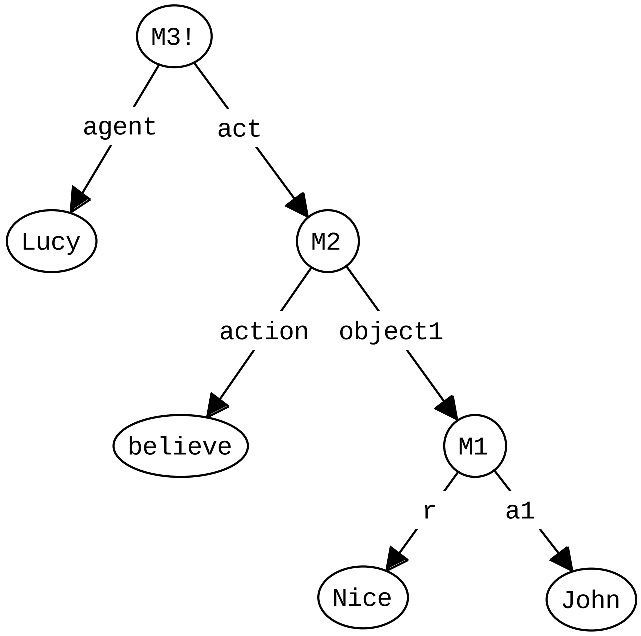


Figure 6.2: Representing `B(Lucy, Nice(John))` as a SNePS network

to say anything about the network they dominate. Similarly, individual parts of a proposition, for example, the antecedent $\mathsf{Man}(\mathsf{x}) \vee \mathsf{Woman}(\mathsf{x})$, are also represented by unique nodes, for example, `P3`. Such parts of a proposition are important during inference, which will become clear below.

6.1 Basic Top-level Inference

Simulative reasoning is a method in which an agent such as Cassie attributes her own reasoning skills to another agent in order to simulate that agent's reasoning. Thus, in order to describe how SIMBA implements simulative reasoning, we have to first explain the basic mechanism by which Cassie's own reasoning is implemented as top-level inference. We will do so with help of a very simple example which will provide the basis for the more complicated examples to follow.

Suppose that Cassie believes that Fido is a dog, and that all dogs are cute. From that she should be able to infer that Fido is cute. Below we present a trace from an actual SNePSLOG interaction that demonstrates how we can provide Cassie with these initial beliefs, and how she can then carry out such an inference in order to answer a question. The trace is cut up into various segments to allow room for explanations.

In the example trace below, and in all SNePSLOG traces to follow, user input at the SNePSLOG prompt (a ':') is shown in a **bold** font. Every line of user input is followed by one of more lines of SNePSLOG response which are shown in standard `typewriter` font. This and all other traces are results from actual runs of the implementation of SIMBA. The traces were modified only slightly for formatting purposes, and once in a while uninteresting portions were deleted to save space. Such deletions are indicated by a series of dots.

We start out by issuing the `clearkb` command to completely clear the underlying SNePS knowledge base, and thus empty Cassie's "mind". The `clearkb` command also creates an empty top-level reasoning context named "Cassie":

: **clearkb**

```
                                                    ...opened Cassie
Knowledge Base Cleared
```

SIMBA uses a special graphics package to display reasoning contexts and their evolution during inference. These graphical reasoning contexts have almost the same format as the reasoning contexts

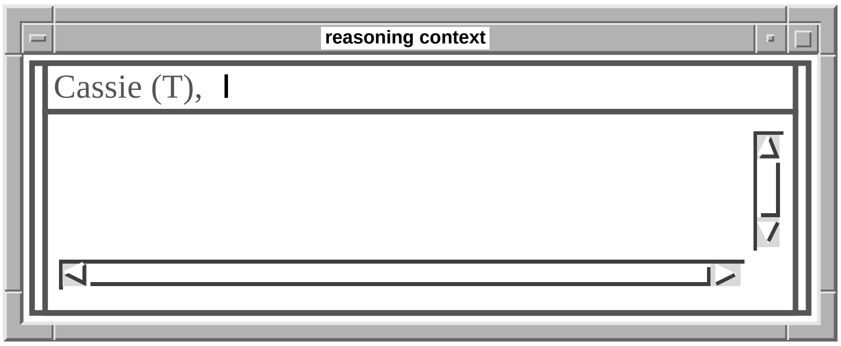


Figure 6.3: The empty top-level reasoning context

used in **SL**. The only difference is that they have window borders and horizontal and vertical scroll bars. The empty context created by `clearkb` is shown in Figure 6.3. To synchronize the SNePSLOG trace with the creation and evolution of graphical reasoning contexts, annotations are provided on the right margin of the trace. For example, the annotation "`...opened Cassie`" right after the `clearkb` command indicates that a graphical reasoning context with name "Cassie" was created.

Now we have to give Cassie some initial beliefs. A new belief is entered into Cassie's mind by typing an appropriate proposition at the SNePSLOG prompt followed by a dot. The dot is an assertion operator that asserts the preceding proposition as an otherwise unjustified belief hypothesis in the *current reasoning context*. Every SNePSLOG command such as assertion, deduction, etc., is carried out in a reasoning context, which, if not explicitly specified, is assumed to be the current reasoning context. The `clearkb` command set the current reasoning context to be the top-level Cassie context. Since throughout most user commands will be carried out in the top-level Cassie context, we will almost never explicitly specify a context with a command:

: **Dog(Fido).**

```
                                                    ...completed step 1
Dog(Fido)
```

: **all(x)(Dog(x) => Cute(x)).**

```
                                                    ...completed step 2
all(x)(Dog(x) => Cute(x))
```

First, we told Cassie that Fido is a dog, and then that all dogs are cute by means of a universally quantified proposition. An annotation of the form "`...completed step N`" is printed whenever a newly asserted or derived sentence is added to a graphical reasoning context. After every operation

SNePSLOG prints its *result*, which in the case of assertion is always the entered proposition. Results are explained in more detail below.

What the top-level Cassie context looks like after these two initial assertions is shown in Figure 6.4. The way reasoning steps are displayed in graphical reasoning contexts is also almost

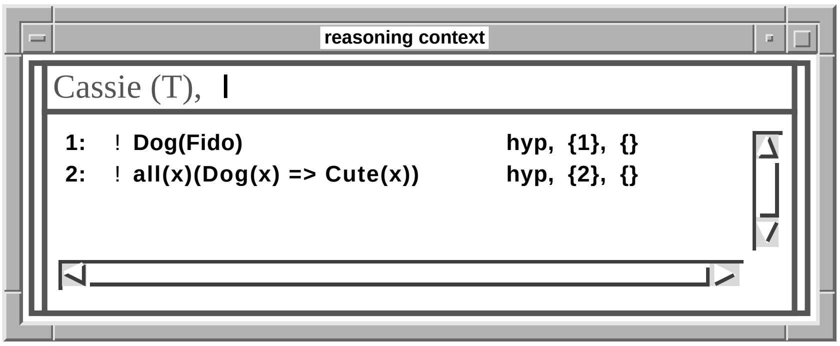


Figure 6.4: The top-level reasoning context after the initial assertions

identical with the way they were displayed in **SL**. The only differences are that for technical reasons step numbers are shown inside the context box instead of outside, and that no explanations are shown. The explanation for every reasoning step is provided by the associated SNePSLOG trace. Since the initial two assertions constitute otherwise unjustified belief hypotheses, they get a hypothesis support which is indicated by the `hyp` origin tag and the singleton hypothesis origin set that contains only the hypothesis itself. As always, we use numbers as aliases for propositions in origin sets to save space. The number of the reasoning step in which a proposition appeared for the first time is used as its alias.

SNePSLOG operations such as assertion or deduction have a *result*. The result is the set of newly asserted or derived propositions. To report such a result back to the user, SNePSLOG applies a generator grammar to the SNePS network that represents the proposition, and then prints the generated string as a response. In the case of assertion the result is always the asserted proposition itself; thus, in most cases the SNePSLOG response will be identical to what was entered by the user. The reason this is not true in all cases is that there are some syntactically different input strings that get translated into the same SNePS network. In that case SNePSLOG chooses a canonical representation during the generation step. This canonical representation might then differ from what was actually entered.

Now we are ready to ask Cassie whether Fido is cute. We do so by typing the proposition followed by a question mark (since we did not specify a context with the question, it is assumed to be the current reasoning context, which is the top-level Cassie context). Since Cassie does not yet have an explicit belief to that effect, she has to use inference to answer the question. During inference special trace messages illustrate the current state of her reasoning:

: **Cute(Fido)?**

```
I wonder if Cute(Fido)
holds in the top-level context
```

```
I wonder if Dog(Fido)
holds in the top-level context
```

When a proposition followed by a question mark is entered into SNePSLOG, it issues a backward inference request to the SNePS Inference Package (SNIP) for that proposition. SNIP supports three principle modes of inference: Forward inference, backward inference, and bi-directional inference, which is a special form of forward inference in which preceding backward inference is used to focus subsequent forward inference. In the examples to follow we will only make use of backward inference; thus, we will not say anything more about the other two inference modes. These and other features of SNIP are described in more detail in [McKay and Shapiro, 1981; Shapiro *et al.*, 1982; Hull, 1986; Choi, 1993].

When SNIP gets the request to find out whether Cute(Fido) is believed in the top-level context, it first finds the SNePS node that represents that proposition (or builds it if it did not exist already), and then sends an "are you believed" request to that node. The request activates a special process associated with the node, and all further inference is carried out by node processes which activate and initiate processes of other nodes by sending request and report messages to them. An underlying simulated multi-processing system called MULTI [McKay and Shapiro, 1980] executes these processes in a pseudo-parallel fashion. Often we will sloppily talk about a certain action of a node or proposition, when in fact we mean the action carried out by its associated process.

When the SNePS node representing Cute(Fido) receives the request asking whether it is believed in the top-level context, it first checks whether it already is believed there. It does so by comparing its origin sets with the set of hypotheses and assumptions that constitute the current belief state of the top-level context. If its support renders it believable in that context (basically, if its origin sets are subsets of the hypotheses and assumptions already believed at the top level), then it will report that it is believed. Since Cute(Fido) does not yet have any support associated with it, it is not believed in the top-level context. Thus, as a next step, it tries to find out whether it can be derived from other propositions that are believed already.

SNIP uses a form of natural deduction to carry out inference. It is able to perform a wide variety of interesting inferences, but it is not a complete inference procedure. Thus, not every inference sequence justified by the logic **SL** will be found by SNIP. SNIP operates by categorizing propositions into *rules* and *non-rules*. Rules are propositions that contain some logical connective; all others are non-rules. For example, the universally quantified proposition all(x)(Dog(x) => Cute(x)) is a rule, while Cute(Fido) is a non-rule. Rules should be understood as *domain rules*, since they express some regularity about the underlying domain. They should not be confused with *inference rules*, which are the rules of the underlying logic that govern the manipulation of logical connectives used in domain rules.

The principle mechanism used by SNIP to derive some proposition p with backward inference, is to find some rule whose consequent is unifiable with p, and then to find out whether the rule itself and all its antecedents are believed. If so it will report that the consequent of the rule is believed. Checking the belief status of the rule and its antecedents might in turn trigger more inference.

The notions of *antecedent* and *consequent* used by SNIP are more general than their usual definition as parts of an entailment. For example, the individual conjuncts of a conjunction $p \wedge q$ are viewed as consequents, since, if the conjunction is believed, the conjuncts can be derived by applying the inference rule of and-elimination ($\wedge$E). On the other hand, the conjuncts are viewed as antecedents in the case where SNIP tries to find out whether the conjunction is believed, since each conjunct has to be believed in order to introduce the conjunction by applying the inference rule of and-introduction ($\wedge$I). The proper application of the inference rules of the logic for each type of domain rule (or logical connective) is handled by specialized functions called *rule handlers*.

Applying this to our example, we can explain the initial two inference trace messages shown above. When the SNePS node representing Cute(Fido) determined that it was not believed already, it first issued the message "`I wonder if Cute(Fido) holds...`", and then it tried to find out the belief status of Cute(Fido) via backward inference. To do that it found all other SNePS nodes that it was unifiable with, and then it sent a request to each of those nodes asking whether it was believed under the substitution determined by the unification operation. The only interesting such node found was the one representing the consequent Cute(x) of the universally quantified entailment. It was unifiable with Cute(Fido) by substituting Fido for the variable x. When Cute(x) received the request that asked whether it was believed under the substitution x/Fido, it first determined that it was in consequent position of the rule all(x)(Dog(x) => Cute(x)), and it then sent a request to the rule asking "can you find out whether I am believed under the substitution x/Fido?" To find that out, the SNePS node representing the rule first checked its own belief status, and since it could determine that it was believed, it then asked its antecedent, at which time the message "`I wonder if Dog(Fido) holds...`" was generated.

Now the node representing the antecedent Dog(x) tries to find nodes that are unifiable with it under the additional constraint that the substitution x/Fido has to hold. There is only one such node in the network, which is the one that represents the proposition Dog(Fido). When that node receives the request sent by Dog(x), it immediately reports back that it is believed, and the following inference trace message is generated:

```
I know Dog(Fido)
```

The rule antecedent Dog(x) then reports to the rule, and the rule handler for entailments applies the inference rule of Modus Ponens ($\Rightarrow$E) (combined with universal instantiation) and generates an inference trace message that explains the derivation of Cute(Fido). A report is then sent to the rule consequent, and finally the proposition Cute(Fido) receives the report that it is believed in the top-level context. Since at that point a reasoning step is completed and a new sentence got added to the graphical top-level reasoning context, an appropriate "`...completed step N`" annotation is printed. The newly derived proposition is then returned as the result of the deduction, and a CPU time message reports how many seconds it took to derive the result:

```
Since all(x)(Dog(x) => Cute(x))
and Dog(Fido)
I infer Cute(Fido)
                                                  ...completed step 3
 Cute(Fido)

 CPU time : 0.27
```

What the top-level Cassie context looks like after the example is completed is shown in Figure 6.5. The support of the newly derived proposition has a `der` origin tag, since it is derived, and its hypothesis origin set contains the aliases of the two propositions on which the derivation was based.

The new support of Cute(Fido) gets permanently stored with it; thus, if we ask the same question a second time, we get an answer immediately without having to go through the whole inference sequence again:

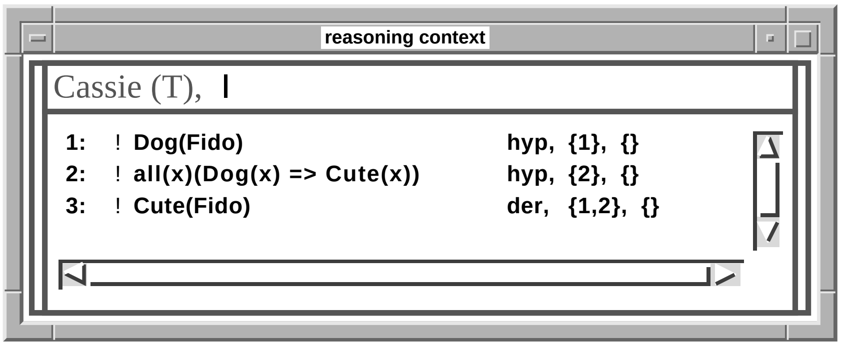


Figure 6.5: The final state of the top-level reasoning context

: **Cute(Fido)?**

```
I know Cute(Fido)

 Cute(Fido)

 CPU time : 0.03
```

6.2 Simple Simulation

In the next example Cassie simulates another agent's reasoning with an entailment. This more or less constitutes the simplest, non-trivial case of simulative reasoning. We will use it as a vehicle to describe the principle mechanism by which propositions are imported into simulation contexts, simulation is carried out, and then results are exported into the parent context as newly derived belief propositions.

Since in this example and in most of the ones to follow we are mainly interested in demonstrating how simulative reasoning interacts with Cassie's own reasoning, but not so much in the various forms of reasoning that could occur during a simulation, we keep things simple by doing the following: First, we will use propositional constants such as P, Q, R, etc., wherever possible, to abstract from any complexities that might arise due to the structure of "real" propositions. Second, we will use reasoning with an entailment by applying Modus Ponens (⇒E) as the prototypical simulative reasoning step.

Both simplifications introduce place holders that abstract from the complexity of the things they stand for. Propositional constants are used instead of arbitrarily complex propositions, and applications of Modus Ponens are used instead of arbitrarily complex reasoning sequences. These simplifications are not at all restrictions of SIMBA, which we will demonstrate later with the help of examples that do not make use of them.

Here is the initial setup of Cassie's top-level context (the final state of the reasoning contexts is shown in Figure 6.7):

: **clearkb**

```
...opened Cassie
```

	Cassie ($\top$), I			
1	$!\,\mathsf{B}(\mathsf{Lucy},\mathsf{P})$,	hyp,	$\{1\}$, $\{\}$	H
2	$!\,\mathsf{B}(\mathsf{Lucy},\mathsf{P}\Rightarrow\mathsf{Q})$,	hyp,	$\{2\}$, $\{\}$	H
				open Lucy
6	$!\,\mathsf{B}(\mathsf{Lucy},\mathsf{Q})$,	sim,	$\{1,2\}$, $\{6\}$	BI 5

	Lucy (Cassie), Lucy, $3/\{1\}\{\}, 4/\{2\}\{\}$			
3	$!\,\mathsf{P}$,	hyp,	$\{3\}$, $\{\}$	SH 1
4	$!\,\mathsf{P}\Rightarrow\mathsf{Q}$,	hyp,	$\{4\}$, $\{\}$	SH 2
5	$!\,\mathsf{Q}$,	sim,	$\{3,4\}$, $\{\}$	$\Rightarrow$E 3,4

Figure 6.6: Simple simulation in **SL**

```
 Knowledge Base Cleared

: B(Lucy, P).
                                                  ...completed step 1

 B(Lucy, P)

: B(Lucy, P => Q).
                                                  ...completed step 2

 B(Lucy, P => Q)
```

Cassie's only beliefs are two beliefs about Lucy's beliefs, that is, that Lucy believes P and $\mathsf{P}\Rightarrow\mathsf{Q}$. From that Cassie should be able to infer that Lucy believes Q by way of simulative reasoning. Figure 6.6 shows how that can be done according to the logic **SL**. Here is how Simba handles it:

```
: B(Lucy, Q)?

I wonder if B(Lucy, Q)
holds in the top-level context
                                                  ...opened Lucy

I wonder if Q
holds in the simulation context Lucy

I wonder if P => Q
holds in the simulation context Lucy
```

The main difference between the standard version of SNIP and the one used by Simba is that the Simba version knows how to reason with belief propositions. In the standard version inference would have terminated unsuccessfully right after the initial "`I wonder if B(Lucy, Q) ...`" message was issued, since there is no other information available from which $\mathsf{B}(\mathsf{Lucy},\mathsf{Q})$ could have been derived. The Simba version, however, knows that one way to derive a belief proposition $\mathsf{B}(a,p)$, is to derive its object proposition p in a simulation context for agent a, and then as a consequence of the simulation result p introduce $\mathsf{B}(a,p)$ in the parent context of the simulation context.

Thus, right after the initial trace message is issued, SIMBA opens a new, empty reasoning context called "Lucy", and then sends a request to the SNePS node representing Q asking whether it is believed in the Lucy context. Since Q is not yet believed there (the Lucy context is still empty), it tries to find other propositions from which it can be derived. In this case this is even simpler than in the example presented in Section 6.1, since there are no variables involved. Instead of having to find other propositions that it is unifiable with, the node representing Q immediately finds out that it is in consequent position of the entailment $\mathsf{P} \Rightarrow \mathsf{Q}$. Checking that could be done very efficiently, because SNePS never builds two different nodes with identical structure; hence, the node representing Q in the proposition $\mathsf{B}(\mathsf{Lucy}, \mathsf{Q})$ is identical with the node representing the consequent of the entailment $\mathsf{P} \Rightarrow \mathsf{Q}$. All the node representing Q had to do to find out whether it was in consequence position of a rule, was to check whether it was at the tail end of a certain consequence arc.

As a next step Q sends a request to the rule $\mathsf{P} \Rightarrow \mathsf{Q}$ asking "am I believed in the Lucy context?" Now we have a case that is different from the way the rule was used in the basic inference example presented before. When $\mathsf{P} \Rightarrow \mathsf{Q}$ receives the request from its consequent, it first checks whether it itself is believed. But since nothing at all is yet believed in the Lucy context, it certainly is not either. Therefore, before it can ask whether its antecedent is believed, it needs to find out whether itself can be derived from other propositions, which gave rise to the message "`I wonder if P => Q ...`".

What we need at this point, is to introduce $\mathsf{P} \Rightarrow \mathsf{Q}$ as a simulation hypothesis into the Lucy context, since $\mathsf{B}(\mathsf{Lucy}, \mathsf{P} \Rightarrow \mathsf{Q})$ is believed in its parent context. To achieve that, SIMBA treats a belief proposition $\mathsf{B}(a, p)$ as a special type of rule, whose "antecedent" is the complete proposition, and whose "consequent" is its object proposition p. The domain rule character of such a proposition could be paraphrased as "if $\mathsf{B}(a, p)$ is believed then p should be believed in the simulation context for agent a". What makes belief propositions different from standard rules such as entailments, etc., is that they reach across reasoning contexts. In a way, they are a special case of the *lifting rules* used in Guha's formalization of contexts [Guha, 1991].

This explains the next step: When $\mathsf{P} \Rightarrow \mathsf{Q}$ tries to find out whether it is derivable from any other propositions, it determines that it is in consequent position of a rule, namely, the proposition $\mathsf{B}(\mathsf{Lucy}, \mathsf{P} \Rightarrow \mathsf{Q})$, and, hence, it sends a request to that rule asking the familiar "am I believed in the Lucy context?" When $\mathsf{B}(\mathsf{Lucy}, \mathsf{P} \Rightarrow \mathsf{Q})$ receives the request, it first checks whether its agent term is identical with the agent term of the simulation context from which the request originated, and then it checks whether it is believed in the parent context of that simulation context. Since in our example both tests succeed, it sends a report back to $\mathsf{P} \Rightarrow \mathsf{Q}$ which results in the following inference trace message:

```
Since B(Lucy, P => Q)
holds in the top-level context
I will assume that Lucy's belief P => Q
holds in the simulation context Lucy
                                              ...completed step 3
```

At that point $\mathsf{P} \Rightarrow \mathsf{Q}$ gets introduced as a new simulation hypothesis into the Lucy context, and an appropriate origin set translation is recorded which is shown in the top of the context box (see Figure 6.7 for the final state of the reasoning contexts). Since we have completed another reasoning step, an appropriate annotation was printed with it.

Now $\mathsf{P} \Rightarrow \mathsf{Q}$ knows that it is believed in the Lucy context, hence, it can ask its antecedent P whether it is believed there also. With a similar sequence of steps P gets introduced into the Lucy context, which then justifies the application of Modus Ponens to derive Q:

```
I wonder if P
holds in the simulation context Lucy

Since B(Lucy, P)
holds in the top-level context
I will assume that Lucy's belief P
holds in the simulation context Lucy
                                                    ...completed step 4
Since P => Q
and P
I infer Q
                                                    ...completed step 5
```

We started out by asking whether $\mathsf{B(Lucy, Q)}$ was believed, which triggered the request for Q in the simulation context Lucy. Since Q is now successfully derived in this context, it reports that back to the requesting proposition $\mathsf{B(Lucy, Q)}$, which leads to its introduction as a derived belief proposition into the top-level context. At that point inference is complete, and the requested proposition gets reported as a result:

```
Since Q
was derived in the simulation context Lucy
under the assumption that Lucy believes
   P
and P => Q
I infer B(Lucy, Q)
                                                    ...completed step 6
 B(Lucy, Q)

 CPU time : 1.38
```

The final state of the reasoning contexts is shown in Figure 6.7. The graphical contexts are almost identical with the reasoning contexts used in the **SL** derivation in Figure 6.6. The only difference is that due to the nature of backward inference the simulation hypotheses were introduced into the Lucy context in a different order.

The introduced belief proposition is of course a simulation assumption, which is indicated by its non-empty assumption origin set, and by its only plausible believability. To emphasize the assumptional character of such propositions, they are displayed in gray instead of black in graphical reasoning contexts. Lesser degrees of believability are indicated by lighter shades of gray. Similarly, elements of assumption origin sets are also always written in gray, but only one shade of gray is used there.

We can get a summary of all reasoning steps with the SNePSLOG command `list-all-steps`. It displays the believabilities of individual sentences relative to the context that was supplied as a first argument. This allows us to verify that none of the sentences derived in the simulation context are accidentally believed in the top-level context, and vice versa. In ASCII notation '`$`' is

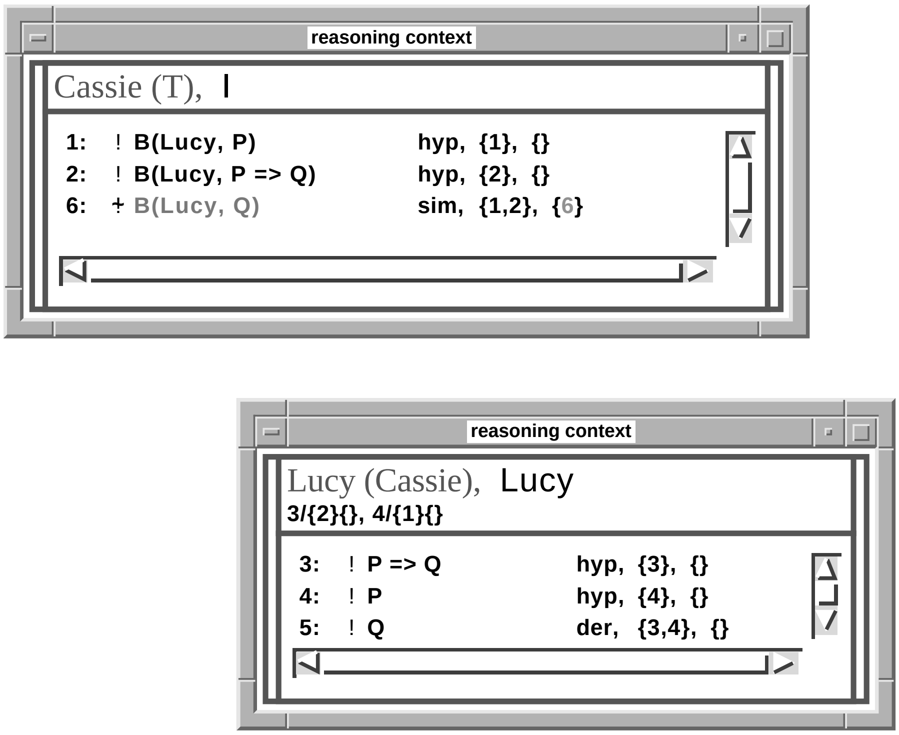


Figure 6.7: Simple simulation in SIMBA

used as the symbol for plausible believability, '?' for conceivable believability, and blank space for unbelievable propositions. The second argument to `list-all-steps` makes it print supports with each step:

: list − all − steps Cassie t

```
1:   ! B(Lucy, P)        hyp, {1}, {}
2:   ! B(Lucy, P => Q)   hyp, {2}, {}
3:     P => Q            hyp, {3}, {}
4:     P                 hyp, {4}, {}
5:     Q                 der, {3,4}, {}
6:   $ B(Lucy, Q)        sim, {1,2}, {6}
```

: list − all − steps Lucy t

```
1:     B(Lucy, P)        hyp, {1}, {}
2:     B(Lucy, P => Q)   hyp, {2}, {}
3:   ! P => Q            hyp, {3}, {}
4:   ! P                 hyp, {4}, {}
5:   ! Q                 der, {3,4}, {}
6:     B(Lucy, Q)        sim, {1,2}, {6}
```

6.3 Multi-Level Simulation

The next example demonstrates that the reasoning scheme used for the simple, one-level simulation described in the previous section, works just as well for simulations that take place at multiple levels of nesting. The only difference between this and the previous example is that the object propositions P and P ⇒ Q are nested three levels deep as Lucy's beliefs of Sally's beliefs of John's beliefs. Here is the initial setup of the top-level Cassie context (the final state of the reasoning contexts is shown in Figure 6.8):

```
: clearkb
                                                  ...opened Cassie
 Knowledge Base Cleared

: B(Lucy, B(Sally, B(John, P))).
                                                  ...completed step 1
 B(Lucy, B(Sally, B(John, P)))

: B(Lucy, B(Sally, B(John, P => Q))).
                                                  ...completed step 2
 B(Lucy, B(Sally, B(John, P => Q)))
```

When we now ask whether B(Lucy, B(Sally, B(John, Q))) is believed, SIMBA tries to infer whether B(Sally, B(John, Q)) is believed in the Lucy context, but that context is still empty and we asked for a belief proposition, hence, another simulation of Sally's reasoning is started from the Lucy context, which in turn triggers a simulation of John's reasoning from the Sally context. Here is the inference trace of these initial requests in the various contexts:

```
: B(Lucy, B(Sally, B(John, Q)))?

I wonder if B(Lucy, B(Sally, B(John, Q)))
holds in the top-level context
                                                  ...opened Lucy
I wonder if B(Sally, B(John, Q))
holds in the simulation context Lucy
                                                  ...opened Lucy/Sally
I wonder if B(John, Q)
holds in the simulation context Lucy/Sally
                                                  ...opened Lucy/Sally/John
I wonder if Q
holds in the simulation context Lucy/Sally/John
```

SIMBA automatically creates names for simulation contexts that reflect the current stack of agents simulating each other's reasoning. This makes it easier to follow the inference trace, and it insures that no two reasoning contexts wind up with the same name. We will, however, always use the name of the principal agent of a simulation context to refer to it, if that is possible without creating ambiguity. For example, in the following we will always refer to the John context, instead of the Lucy/Sally/John context.

So far all but the top-level context are still empty. Now Q finds itself in consequent position of the rule P ⇒ Q and sends a request to it. The rule is not yet believed in the John context; hence, it sends a request to the belief proposition $\mathsf{B}(\mathsf{John}, \mathsf{P} \Rightarrow \mathsf{Q})$ in the Sally context, which in turn triggers a request chain that percolates all the way out to the top-level context. There, finally, we have a proposition that is believed, which leads to a chain of imports all the way back to the John context. Here are the associated steps as they are carried out by SIMBA:

```
I wonder if P => Q
holds in the simulation context Lucy/Sally/John

I wonder if B(John, P => Q)
holds in the simulation context Lucy/Sally

I wonder if B(Sally, B(John, P => Q))
holds in the simulation context Lucy

Since B(Lucy, B(Sally, B(John, P => Q)))
holds in the top-level context
I will assume that Lucy's belief B(Sally, B(John, P => Q))
holds in the simulation context Lucy
                                                  ...completed step 3
Since B(Sally, B(John, P => Q))
holds in the simulation context Lucy
I will assume that Sally's belief B(John, P => Q)
holds in the simulation context Lucy/Sally
                                                  ...completed step 4
Since B(John, P => Q)
holds in the simulation context Lucy/Sally
I will assume that John's belief P => Q
holds in the simulation context Lucy/Sally/John
                                                  ...completed step 5
```

Now $\mathsf{P} \Rightarrow \mathsf{Q}$ is believed in the John context, thus, it can ask for its antecedent P. This sets off a similar request and report chain that ends in the introduction of P as a simulation hypothesis in the John context:

```
I wonder if P
holds in the simulation context Lucy/Sally/John

I wonder if B(John, P)
holds in the simulation context Lucy/Sally

I wonder if B(Sally, B(John, P))
holds in the simulation context Lucy

Since B(Lucy, B(Sally, B(John, P)))
holds in the top-level context
I will assume that Lucy's belief B(Sally, B(John, P))
holds in the simulation context Lucy
                                                  ...completed step 6
```

```
Since B(Sally, B(John, P))
holds in the simulation context Lucy
I will assume that Sally's belief B(John, P)
holds in the simulation context Lucy/Sally
                                                         ...completed step 7
Since B(John, P)
holds in the simulation context Lucy/Sally
I will assume that John's belief P
holds in the simulation context Lucy/Sally/John
                                                         ...completed step 8
```

Finally, we can actually simulate John's reasoning and derive Q. That result is then reported from context to context all the way out to the top level:

```
Since P => Q
and P
I infer Q
                                                         ...completed step 9
Since Q
was derived in the simulation context Lucy/Sally/John
under the assumption that John believes
    P
and P => Q
I infer B(John, Q)
                                                         ...completed step 10
Since B(John, Q)
was derived in the simulation context Lucy/Sally
under the assumption that Sally believes
    B(John, P)
and B(John, P => Q)
I infer B(Sally, B(John, Q))
                                                         ...completed step 11
Since B(Sally, B(John, Q))
was derived in the simulation context Lucy
under the assumption that Lucy believes
    B(Sally, B(John, P))
and B(Sally, B(John, P => Q))
I infer B(Lucy, B(Sally, B(John, Q)))
                                                         ...completed step 12
 B(Lucy, B(Sally, B(John, Q)))

 CPU time : 5.29
```

The final state of all reasoning contexts involved in this example is shown in Figure 6.8. Note, that we carried out a simulation in all contexts but the John context, and that the simulations carried out in the Cassie and the Lucy context required in turn simulations to be carried out at deeper levels.

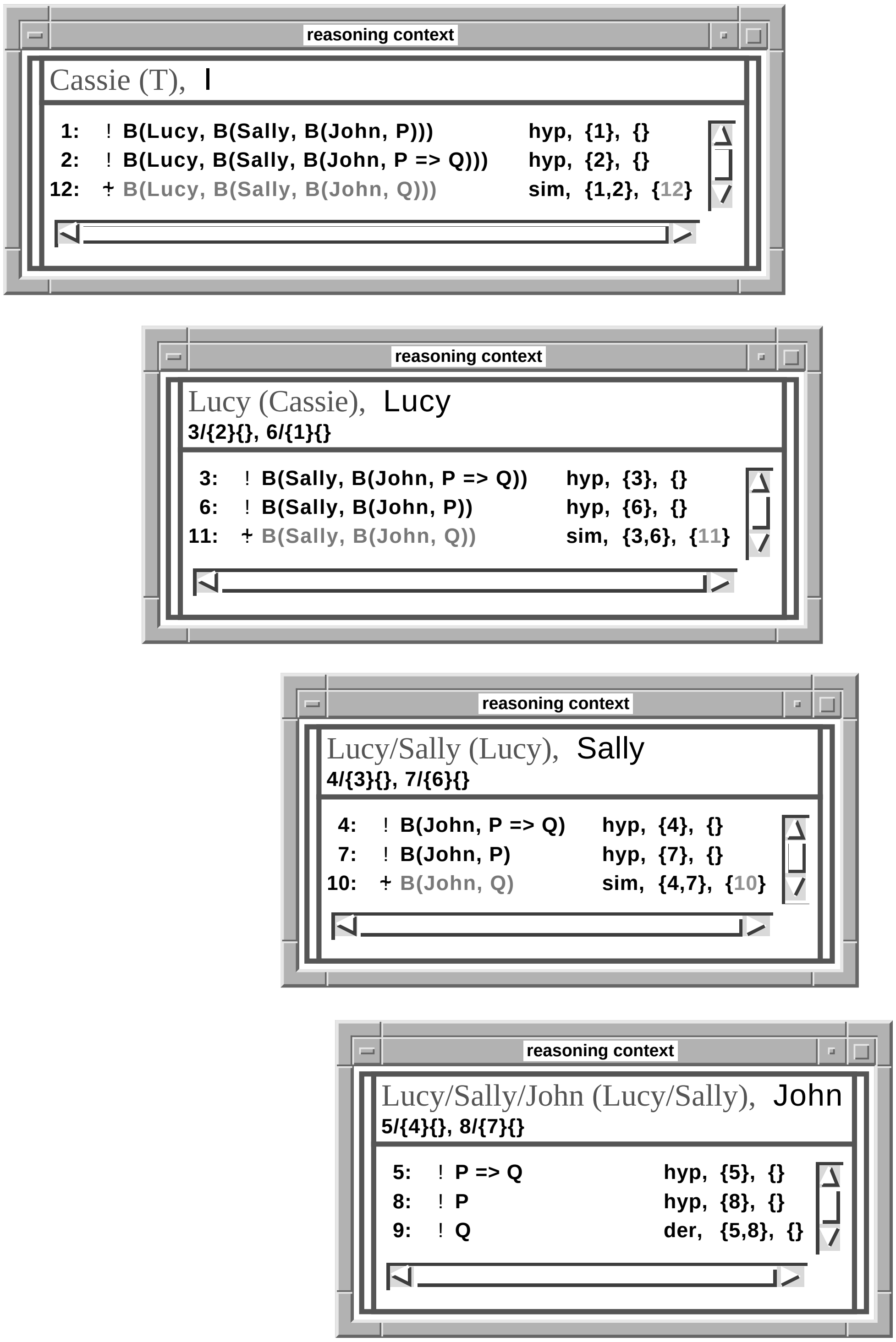


Figure 6.8: Nested simulation going three levels deep

6.4 Meta-Reasoning and Group Belief

The simulative reasoning examples presented so far were both very simple. All that was really necessary even in the multi-level case, was to unwrap object propositions from their respective belief propositions, import the unwrapped propositions into the appropriate simulation contexts, carry out the simulation, and then again wrap the results inside proper belief propositions and export them to the various parent contexts. Thus, instead of the somewhat complicated request and report scheme that imports and exports propositions in and out of contexts lazily on demand, one could have used a simpler, "greedy" scheme, where every time a belief proposition gets asserted its object proposition get asserted automatically with it in the appropriate simulation context.

There are at least two important scenarios in which this simple scheme would fail: *Meta-reasoning* and *hypothetical simulation*. What we call meta-reasoning is the activity to derive not yet explicitly available belief propositions in a simulator or meta-context, in order to make their object propositions available in a simulation context. The example below will show a situation in which that is necessary. Hypothetical simulation is needed to answer questions such as "suppose Lucy believes Fido is a dog; would she then believe that he is cute?" An example for that will be given in the next section.

The multi-level simulation presented before already exhibited some trivial form of meta-reasoning. It occurred, for example, when the proposition $\mathsf{P} \Rightarrow \mathsf{Q}$ asked whether $\mathsf{B(John, P \Rightarrow Q)}$ was believed in the Sally context. Since that proposition was not yet believed in that context, reasoning was necessary to make it available there. That reasoning was admittedly simple, and the greedy approach described above would have made it obsolete.

One situation that can be handled rather elegantly with meta-reasoning, but cannot be handled well with the greedy approach, is the representation and use of group belief (or knowledge). For example, we might want to tell Cassie that all philosophers believe that Nietzsche wrote Thus Spake Zarathustra.[2] A way to do so would be to give her the belief

$$\forall \mathsf{a}\ \mathsf{Philosopher(a)} \Rightarrow \mathsf{B(a, Author(Nietzsche, ZarathustraBook))}$$

and then Cassie should be able to use that whenever she simulates the reasoning of a philosopher. Obviously, the greedy approach would fail here, because it would amount to carrying out full forward inference any time a new belief enters Cassie's mind, just in order to determine whether the new belief qualified any of the currently known agents as a philosopher.

Here is a simple example that shows how meta-reasoning is handled rather straightforwardly by SIMBA. Let us first give Cassie a set of initial beliefs (the final state of the reasoning contexts is shown in Figure 6.9):

```
: clearkb
                                                ...opened Cassie
Knowledge Base Cleared

: Phil(Lucy).
                                                ...completed step 1
Phil(Lucy)
```

[2]The assumption that there is a proposition all philosophers agree on might be construed by some as slightly unrealistic.

: **all(a)(Phil(a) => B(a, P)).**

```
                                                        ...completed step 2
 all(a)(Phil(a) => B(a, P))
```

: **B(Lucy, P => Q).**

```
                                                        ...completed step 3
 B(Lucy, P => Q)
```

Thus, we told Cassie that Lucy is a philosopher, and that all philosophers believe a certain proposition P. We also told her that Lucy believes the simple entailment familiar from the previous examples. Now we are ready to ask her whether Lucy believes Q (some of the request messages that were already discussed before have been deleted to save space):

: **B(Lucy, Q)?**

```
  ..................

Since B(Lucy, P => Q)
holds in the top-level context
I will assume that Lucy's belief P => Q
holds in the simulation context Lucy
                                                        ...completed step 4
I wonder if P
holds in the simulation context Lucy

I wonder if B(Lucy, P)
holds in the top-level context

I wonder if Phil(Lucy)
holds in the top-level context

I know Phil(Lucy)

Since all(a)(Phil(a) => B(a, P))
and Phil(Lucy)
I infer B(Lucy, P)
                                                        ...completed step 5
```

The inference steps following the "`I wonder if P...`" message above are the ones that make this example different from the examples before. Had B(Lucy, P) already been believed in the top-level context, then it would have immediately led to the introduction of P into the Lucy context. But since it is not, it leads to further inference in the top-level context in order to derive it. Since the rule expressing the group belief for philosophers has a matching consequent, a request is sent to it, which chains back to Phil(Lucy), which is believed and in turn leads to the successful derivation of B(Lucy, P). Once that is derived, it reports back to the Lucy context and the rest of the inference proceeds as already seen before:

```
Since B(Lucy, P)
```

```
holds in the top-level context
I will assume that Lucy's belief P
holds in the simulation context Lucy
                                                  ...completed step 6
Since P => Q
and P
I infer Q
                                                  ...completed step 7
Since Q
was derived in the simulation context Lucy
under the assumption that Lucy believes
    P
and P => Q
I infer B(Lucy, Q)
                                                  ...completed step 8
 B(Lucy, Q)

 CPU time : 2.53
```

What the reasoning contexts look like after the example is completed is shown in Figure 6.9. It

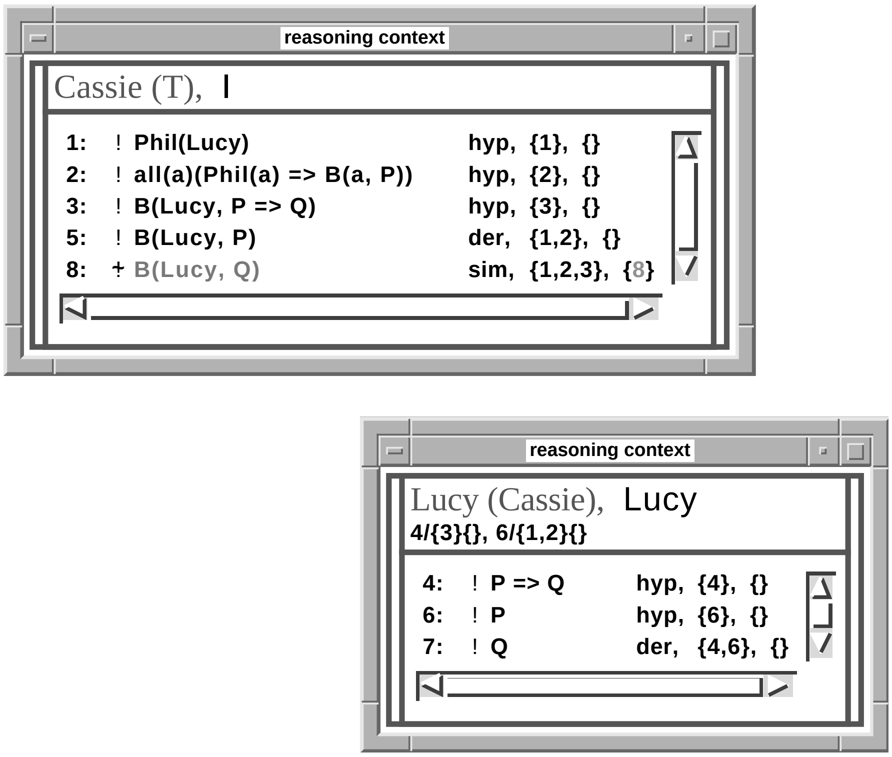


Figure 6.9: Meta-reasoning and group belief

should be pointed out that the form of group belief used here is not *mutual*, in the sense that every philosopher believes it, and believes that every philosopher believes it, and...; thus, Cassie knows the common belief of philosophers, but (in Cassie's view) Lucy does not. To model Lucy as having the same belief we would have needed to represent it explicitly as $\mathsf{B}(\mathsf{Lucy}, \forall \mathsf{a}\, \mathsf{Phil}(\mathsf{a}) \Rightarrow \mathsf{B}(\mathsf{a}, \mathsf{P}))$, and so on, which then would have enabled Cassie to use it in her simulation of Lucy's simulation of other philosophers.

6.5 Hypothetical Simulation

The second case in which the greedy approach described in Section 6.4 would fail is the case of hypothetical simulation. Once in a while Cassie might need to answer a question such as "if Lucy believed p, would she believe q?" To find that out, Cassie would need to perform hypothetical reasoning.

In SIMBA hypothetical reasoning can be carried out in the form of deriving entailments. Thus, propositions as expressed in the question above would be formalized as $\mathsf{B}(\mathsf{Lucy}, p) \Rightarrow \mathsf{B}(\mathsf{Lucy}, q)$. Here is an example how such an entailment can be derived:

```
: clearkb
                                             ...opened Cassie
 Knowledge Base Cleared

: B(Lucy, P => Q).
                                             ...completed step 1
 B(Lucy, P => Q)

: B(Lucy, P) => B(Lucy, Q)?

I wonder if B(Lucy, P) => B(Lucy, Q)
holds in the top-level context

Let me assume that B(Lucy, P)
                                             ...opened =>-Intro-1
                                             ...completed step 2
```

The only initial belief that Cassie has, is that Lucy believes an entailment. Now we can ask "if she believed the antecedent of that entailment, would she believe the consequent?" When SIMBA is asked to derive an entailment, it assumes the antecedent in a hypothetical context and tries to derive the consequent. Thus, right after the question a new hypothetical reasoning context with name `=>-Intro-1` is opened, and $\mathsf{B}(\mathsf{Lucy}, \mathsf{P})$ is assumed in it as an additional hypothesis (see Figure 6.10 for the final state of the reasoning contexts).

The main difference between hypothetical and simulation contexts is that in hypothetical contexts in addition to hypotheses assumed directly within them, everything that is available in their parent context can also be imported into them with the inference rule of reiteration. This lack of an importation barrier is symbolized by only using single vertical lines for the hypothetical context box. In the implementation of SIMBA, reiteration is never carried out explicitly; instead all hypotheses available in the parent context are made available automatically in the hypothetical context. Sentences *derived* and believed in the parent context will automatically be believed in the hypothetical context, since the hypotheses of the hypothetical context form a superset of the ones of its parent context.

When the request for the consequent $\mathsf{B}(\mathsf{Lucy}, \mathsf{Q})$ is triggered in the hypothetical context, a standard simulation ensues. It can be carried out successfully, since in addition to the assumed antecedent of the entailment, the initial belief asserted at the top level is also available. Note, that the Lucy context has the hypothetical context as its parent context, and thus would be different from a Lucy context opened directly from the top level:

```
I wonder if B(Lucy, Q)
holds in the hypothetical context =>-Intro-1
                                                     ...opened =>-Intro-1/Lucy
I wonder if Q
holds in the simulation context =>-Intro-1/Lucy

I wonder if P => Q
holds in the simulation context =>-Intro-1/Lucy

Since B(Lucy, P => Q)
holds in the hypothetical context =>-Intro-1
I will assume that Lucy's belief P => Q
holds in the simulation context =>-Intro-1/Lucy
                                                     ...completed step 3
I wonder if P
holds in the simulation context =>-Intro-1/Lucy

Since B(Lucy, P)
holds in the hypothetical context =>-Intro-1
I will assume that Lucy's belief P
holds in the simulation context =>-Intro-1/Lucy
                                                     ...completed step 4
Since P => Q
and P
I infer Q
                                                     ...completed step 5
Since Q
was derived in the simulation context =>-Intro-1/Lucy
under the assumption that Lucy believes
    P
and P => Q
I infer B(Lucy, Q)
                                                     ...completed step 6
```

At this point SIMBA successfully derived the consequent of the entailment in the hypothetical context. Therefore, it is justified to apply the inference rule of implication introduction ($\Rightarrow$I), and introduce the entailment in the top-level context:

```
Since B(Lucy, Q)
was derived assuming B(Lucy, P)
I infer B(Lucy, P) => B(Lucy, Q)
                                                     ...completed step 7
```

```
B(Lucy, P) => B(Lucy, Q)

CPU time : 3.06
```

The final state of the reasoning contexts is shown in Figure 6.10. Note, that the hypothesis origin

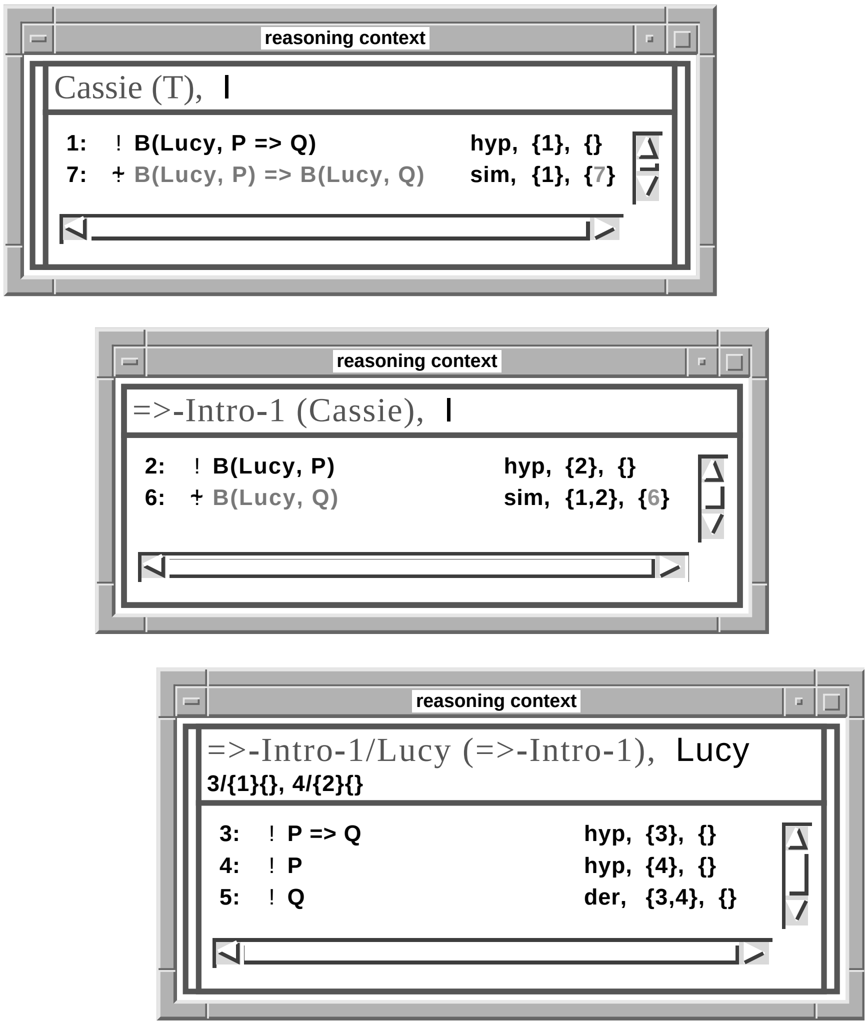


Figure 6.10: Hypothetical simulation

set of the result is a singleton, since the hypothetically assumed antecedent was discharged during implication introduction. However, since the derivation of the entailment was based on a simulation of Lucy's reasoning, the resulting entailment has to be made into a simulation assumption also.

6.6 Simulative Theorem Proving

Even though SNIP is an incomplete inference procedure, it can derive some simple theorems of the logic **SL**. What separates theorems from other derived sentences is that they are not based on any initial hypotheses and assumptions. For that reason, both of their origin sets are empty. Since the belief of a proposition in a particular context is based on whether its origin sets are subsets of the hypotheses and assumptions of the current belief state of a context, theorems will automatically be believed in all reasoning contexts, since the empty set is a subset of every set.

In the example below, we show how SIMBA can derive a theorem in a simulation context in order to answer the question whether some agent believes a particular theorem. While the derived belief proposition will be a simulation assumption, Cassie will automatically come to believe the theorem after it was derived in the simulation context. Hence, when we then ask her whether she believes the theorem herself, she can immediately answer that question without having to derive the theorem again. This time inference is started without giving Cassie any initial beliefs:

: **clearkb**

```
                                                   ...opened Cassie
 Knowledge Base Cleared
```

: **B(Lucy, (P and (P => Q)) => Q)?**

Our question could be paraphrased as whether Cassie believes that Lucy "knows" Modus Ponens. The reason we used quotes here is that even though Cassie can derive a theorem that is an object-level characterization of the inference rule of Modus Ponens, she does not by that become aware of the general applicability of that rule. She simply goes through some inference motions to derive the theorem (the derivation in a sense brings her built-in inference rules to the surface), but if we then asked her whether she believes $(\mathsf{Q} \wedge (\mathsf{Q} \Rightarrow \mathsf{P})) \Rightarrow \mathsf{P}$, she would need to go through the same motions again to derive this syntactic variant of the theorem.

All that aside, here is how the question gets answered. Since we asked for a belief proposition, Cassie as usual starts a simulation of the agent of the belief in order to derive its object proposition. This object proposition is an entailment; hence, in order to derive it Cassie (simulating Lucy) has to use hypothetical reasoning similar to what we described in the previous example. Thus, right after the Lucy context is created, the hypothetical context `=>-Intro-1` is opened from the Lucy context with the antecedent of the entailment assumed in it (the final state of the reasoning contexts is shown in Figure 6.11):

```
I wonder if B(Lucy, (P and (P => Q)) => Q)
holds in the top-level context
                                                   ...opened Lucy
I wonder if (P and (P => Q)) => Q
holds in the simulation context Lucy

Let me assume that P and (P => Q)
                                                   ...opened Lucy/=>-Intro-1
                                                   ...completed step 1
I wonder if Q
holds in the hypothetical context Lucy/=>-Intro-1

I wonder if P => Q
holds in the hypothetical context Lucy/=>-Intro-1

It is the case that P => Q
                                                   ...completed step 2
```

```
I wonder if P
holds in the hypothetical context Lucy/=>-Intro-1

It is the case that P
                                                    ...completed step 3
```

The messages "`It is the case that...`" are printed when individual conjuncts of a conjunction are derived via the inference rule of and-elimination ($\wedge$E). Once these conjuncts are available individually in the `=>-Intro-1` context, `Q` can be derived and the entailment can be introduced in the Lucy context. This simulation result then justifies the introduction of the belief proposition in question at the top level:

```
Since P => Q
and P
I infer Q
                                                    ...completed step 4
Since Q
was derived assuming P and (P => Q)
I infer (P and (P => Q)) => Q
                                                    ...completed step 5
Since (P and (P => Q)) => Q
was derived in the simulation context Lucy
I infer B(Lucy, (P and (P => Q)) => Q)
                                                    ...completed step 6
 B(Lucy, (P and (P => Q)) => Q)

 CPU time : 3.03
```

Figure 6.11 shows the final state of all reasoning contexts. Note that the resulting belief proposition has an empty hypothesis origin set but a non-empty assumption origin set, since it was based on the simulation of Lucy's reasoning. However, if we now ask Cassie whether she believes the theorem in question, she will report so immediately without having to rederive it. This is the case, because it was derived with an empty origin set in the Lucy context; hence, as explained above, it will be believed automatically in all of Cassie's reasoning contexts:

```
: (P and (P => Q)) => Q?

I know (P and (P => Q)) => Q
                                                    ...completed step 7
 (P and (P => Q)) => Q

 CPU time : 0.21
```

Below is a variation on that theme where we first ask Cassie whether she believes the theorem, and then ask her whether Lucy believes it too. In that case the theorem does not have to be rederived during the simulation of Lucy's reasoning; however, the resulting belief proposition still gets a non-empty assumption origin set as required. The inference trace that led to the derivation of the theorem has been deleted, since it is isomorphic to the one shown above:

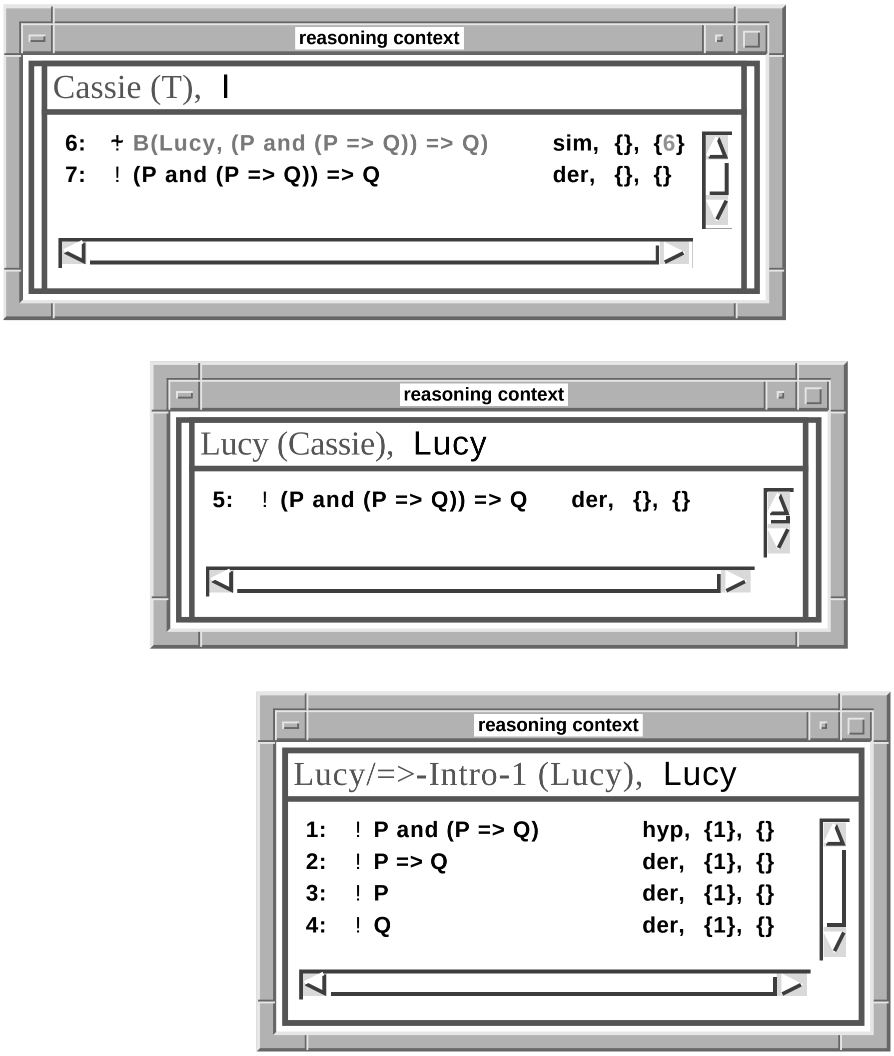


Figure 6.11: Proving a theorem in a simulation context

: **clearkb**

...opened Cassie

```
 Knowledge Base Cleared
```

: **(P and (P => Q)) => Q?**

```
  ...................

Since Q
was derived assuming P and (P => Q)
I infer (P and (P => Q)) => Q
```

...completed step 5

```
 (P and (P => Q)) => Q
```

```
CPU time : 1.81
```

: **B(Lucy, (P and (P => Q)) => Q)?**

```
I wonder if B(Lucy, (P and (P => Q)) => Q)
holds in the top-level context
                                                  ...opened Lucy
I know (P and (P => Q)) => Q
                                                  ...completed step 6
Since (P and (P => Q)) => Q
was derived in the simulation context Lucy
I infer B(Lucy, (P and (P => Q)) => Q)
                                                  ...completed step 7
 B(Lucy, (P and (P => Q)) => Q)

 CPU time : 1.40
```

As yet another twist we can ask whether Lucy's beliefs *entail* the theorem, by using the belief entailment function BE introduced in section 5.5:

: **BE(Lucy, (P and (P => Q)) => Q)?**

```
I wonder if BE(Lucy, (P and (P => Q)) => Q)
holds in the top-level context

I know (P and (P => Q)) => Q

Since (P and (P => Q)) => Q
was derived in the simulation context Lucy
I infer BE(Lucy, (P and (P => Q)) => Q)
                                                  ...completed step 8
 BE(Lucy, (P and (P => Q)) => Q)

 CPU time : 0.34
```

The final state of the reasoning contexts is shown in Figure 6.12. Apart from the belief entailment, the top-level results are the same, but the nesting of the contexts is different from what it was before. This example brings the difference between belief and belief entailment to a point. The derived belief entailment is itself a theorem, since any agent's empty set of beliefs must entail any theorem of **SL** according to our logic. On the other hand, the belief proposition expresses Cassie's belief whether Lucy really believes the theorem in question. That is of course an assumption, since Lucy might not have done the reasoning herself.

6.7 WH-Questions

So far all the questions asked in the various examples were fully ground; that is, we always asked Cassie whether she believed one particular, fully specified proposition. Often, however, it is desir-

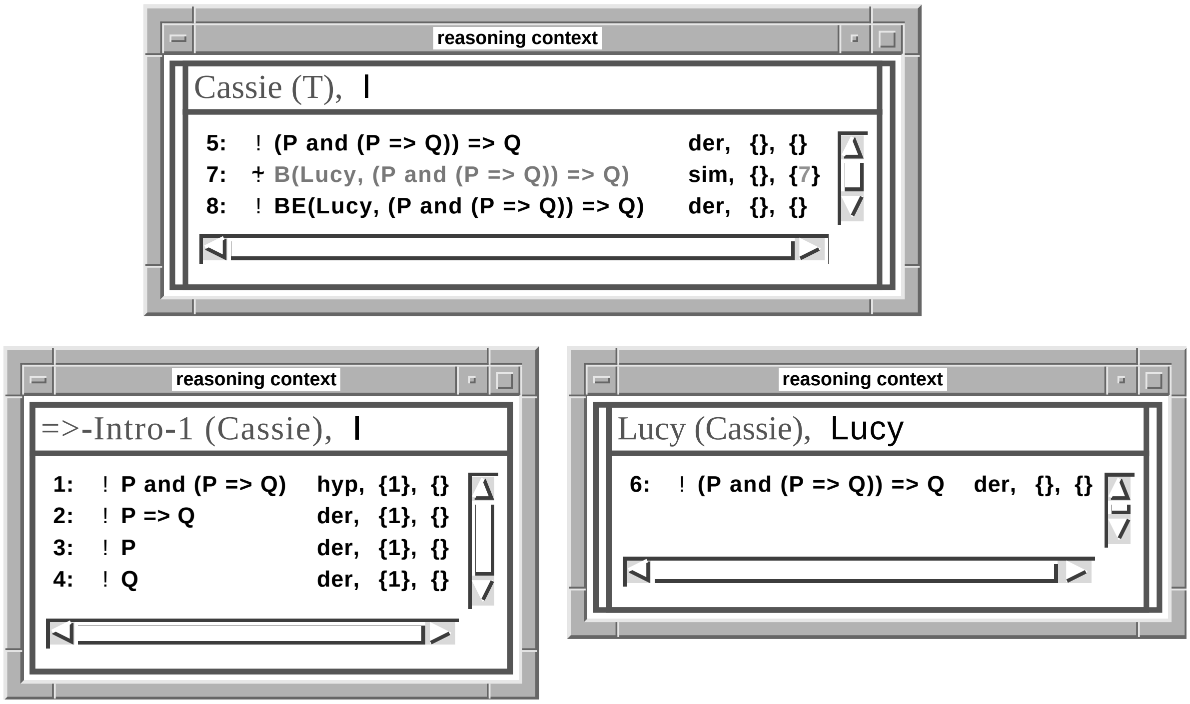


Figure 6.12: Importing a theorem into a simulation context

able to be able to ask an open or WH-question, for example, "who do you believe is smart?", or "who do you believe Lucy believes is smart?" Such questions will contain one ore more variables representing the various WH-words in the question. What we would like as an answer in such a case, is some set of believed propositions that match or are unifiable with the open, underspecified proposition used in the question.

SNePS already comes with a mechanism that handles WH-questions rather naturally. It is basically that mechanism which is used by SIMBA to answer the plain WH-question in the first example below. A second kind of WH-questions, however, requires a somewhat different treatment. These are belief questions that involve an agent variable, for example, "who believes Lucy is rich?" Since simulations of an agent's reasoning can only be carried out once that agent is known, a special mechanism has to be used for open belief questions. The second example below will show how those are handled by SIMBA.

6.7.1 Plain WH-Question

In the first example, we simply ask for all instances of a proposition derivable in a simulation context. What happens within the simulation context to derive those instances is basically the same as what would have happened in a similar reasoning sequence carried out at the top level. The main differences are the additional import and export steps necessary, to move propositions in and out of the simulation context.

Here are Cassie's initial beliefs. All of them are about Lucy, in particular, that Lucy knows of two philosophers Socrates and Plato and that she believes that all philosophers are smart (the final state of the reasoning contexts is shown in Figure 6.13):

```
: clearkb
                                                  ...opened Cassie
 Knowledge Base Cleared

: B(Lucy, Phil(Soc)).
                                                  ...completed step 1
 B(Lucy, Phil(Soc))

: B(Lucy, Phil(Plato)).
                                                  ...completed step 2
 B(Lucy, Phil(Plato))

: B(Lucy, all(a)(Phil(a) => Smart(a))).
                                                  ...completed step 3
 B(Lucy, all(a)(Phil(a) => Smart(a)))
```

Having given Cassie these initial beliefs, we can ask her who she believes Lucy believes is smart. An open question such as this, also called a *pattern*, is given to SNePSLOG by replacing parts of the asked proposition with variables. These variables are symbols prefixed with a question mark, for example, the variable `?who` used below. The question mark is only used to specify variables in the question. In the inference trace, these variables are printed just like all other variables as lower-case symbols without the question mark:

```
: B(Lucy, Smart(?who))?

I wonder if B(Lucy, Smart(who))
holds in the top-level context
                                                  ...opened Lucy
I wonder if Smart(who)
holds in the simulation context Lucy

I wonder if all(a)(Phil(a) => Smart(a))
holds in the simulation context Lucy

Since B(Lucy, all(a)(Phil(a) => Smart(a)))
holds in the top-level context
I will assume that Lucy's belief all(a)(Phil(a) => Smart(a))
holds in the simulation context Lucy
                                                  ...completed step 4
I wonder if Phil(a)
holds in the simulation context Lucy

Since B(Lucy, Phil(Plato))
holds in the top-level context
I will assume that Lucy's belief Phil(Plato)
holds in the simulation context Lucy
                                                  ...completed step 5
Since all(a)(Phil(a) => Smart(a))
```

```
and Phil(Plato)
I infer Smart(Plato)
                                                ...completed step 6

Since Smart(Plato)
was derived in the simulation context Lucy
under the assumption that Lucy believes
    Phil(Plato)
and all(a)(Phil(a) => Smart(a))
I infer B(Lucy, Smart(Plato))
                                                ...completed step 7
```

Right after the question a request for Smart(who) is issued in the Lucy context. What happens from there is very similar to the case where we looked for a ground proposition, for example, the case described in the basic inference example in Section 6.1. What is different, is that when Smart(who) looks for other SNePS nodes that it is unifiable with, it finds the consequent Smart(a) of the quantified rule, and the unification procedure has to make sure that the different variables are properly substituted for each other.

Now, the quantified rule gets introduced into the simulation context in step 4, and it then asks its antecedent Phil(a) whether it is believed without providing any restrictions on the variable. Phil(a) finds exactly two propositions it is unifiable with and sends a request to each of them. The first one that reports back into the Lucy context is Phil(Plato) which, once the reasoning across the rule is complete leads to the derivation of Smart(Plato). That result is then reported to the parent context as B(Lucy, Smart(Plato)) which forms the first derived instance of the question that we asked.

Now a similar sequence is carried out for the second philosopher, and at the end the two derived instances are reported as the result of the question:

```
Since B(Lucy, Phil(Soc))
holds in the top-level context
I will assume that Lucy's belief Phil(Soc)
holds in the simulation context Lucy
                                                ...completed step 8

Since all(a)(Phil(a) => Smart(a))
and Phil(Soc)
I infer Smart(Soc)
                                                ...completed step 9

Since Smart(Soc)
was derived in the simulation context Lucy
under the assumption that Lucy believes
    Phil(Soc)
and all(a)(Phil(a) => Smart(a))
I infer B(Lucy, Smart(Soc))
                                                ...completed step 10

 B(Lucy, Smart(Soc))
 B(Lucy, Smart(Plato))

 CPU time : 4.44
```

What the reasoning contexts look like after the example is complete, is shown in Figure 6.13.

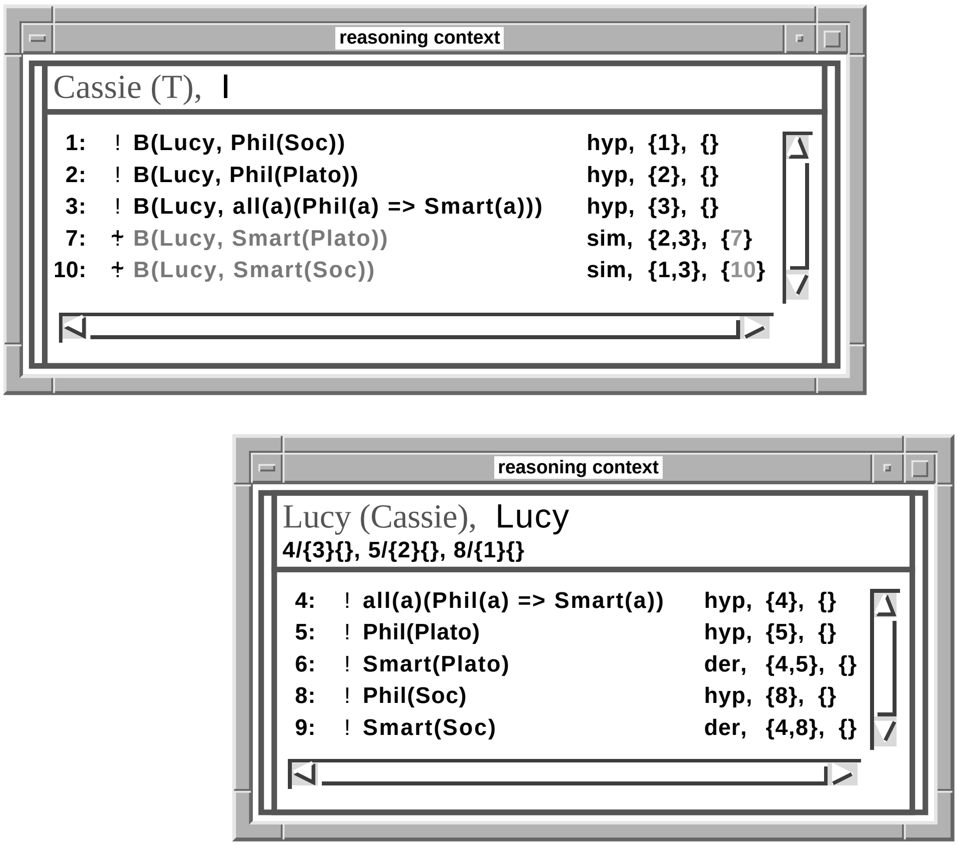


Figure 6.13: Answering a WH-question

6.7.2 Who Believes Something?

The type of WH-question that cannot be handled by the standard SNePS mechanism is when a belief proposition pattern such as $\mathsf{B}(\mathsf{a},\mathsf{P})$ with an agent variable is requested, and the derivation of instances of that pattern needs the simulation of some actual agents in order to derive P.

Here is a run that exemplifies this situation. Suppose that Cassie believes that Socrates and Plato are philosophers, that philosophers believe that they are human, and that philosophers also believe that all humans are mortal. These latter two rules are instances of group belief as discussed in Section 6.4. Finally, suppose Cassie believes that whoever believes of themselves to be mortal is smart (admittedly, a rather crude measure of smartness). Here is how these initial beliefs are formalized (the final state of the reasoning contexts is shown in Figure 6.14):

: **clearkb**

```
                                        ...opened Cassie
Knowledge Base Cleared
```

: **Phil(Soc).**

```
                                        ...completed step 1
Phil(Soc)
```

: **Phil(Plato).**

```
                                              ...completed step 2
 Phil(Plato)
```

: **all(a)(Phil(a) => B(a, Human(a))).**

```
                                              ...completed step 3
 all(a)(Phil(a) => B(a, Human(a)))
```

: **all(a)(Phil(a) => B(a, all(x)(Human(x) => Mortal(x)))).**

```
                                              ...completed step 4
 all(a)(Phil(a) => B(a, all(x)(Human(x) => Mortal(x))))
```

: **all(a)(B(a, Mortal(a)) => Smart(a)).**

```
                                              ...completed step 5
 all(a)(B(a, Mortal(a)) => Smart(a))
```

Note, that there is not a single ground nested belief proposition among these initial beliefs of Cassie. What we want to ask Cassie is who she believes to be smart. The reasoning she has to carry out can be paraphrased like this: Since both Plato and Socrates are philosophers, they believe themselves to be human. Since they also believe that all humans are mortal, they must believe (by simulation) that they are mortal. Since they both believe themselves to be mortal, they must be smart. Here is how SIMBA handles the question:

: **Smart(?who)?**

```
I wonder if Smart(who)
holds in the top-level context

I wonder if B(a, Mortal(a))
holds in the top-level context

I wonder if Mortal(a)
holds in the simulation context *VarAgP9*

I wonder if all(x)(Human(x) => Mortal(x))
holds in the simulation context *VarAgP9*
```

To answer the question Smart(who) SIMBA has to find all instances of the belief proposition pattern B(a, Mortal(a)), since every such instance will qualify the instantiated agent as smart. The standard way to derive a belief proposition such as this is to try to derive its object proposition in a simulation of the reasoning of its agent. But now that agent is a variable, and the inference rules of **SL** do not allow simulations with variable agents (mainly, because they do not allow non-ground proposition patterns as premises).

To handle that situation SIMBA uses the following trick: It opens a dummy simulation context, `*VarAgP9*` in our example, that has a non-ground, variable agent, and it stores the requested pattern that created the context with it. In our example, that pattern is Mortal(a), and the suffix `P9` used in the context name is the SNePS node identifier of the pattern node that represents Mortal(a).

The request for Mortal(a) in the *VarAgP9* context triggers more requests as usual, and these requests branch out to propositions believed in the parent context. Arbitrarily complex inference might have to be carried out in the parent context to derive actual belief propositions that could be used in a simulation. When the first such proposition reports back to the *VarAgP9* context, the rule handler function that handles such reports checks whether the target simulation context has a variable agent or not. If it does it substitutes the agent of the reporting belief proposition which is known, for the variable agent of the request that was stored with *VarAgP9* when that context was opened, and instead of reporting the object proposition of the report to the *VarAgP9* context, it opens a simulation context for the known agent of the reporting proposition and issues the modified request in that context. For example, if the agent of the reporting belief proposition was Plato, then a simulation context for Plato is opened, and a request for Mortal(Plato) is issued in it. This process is repeated whenever a belief proposition sends a report to *VarAgP9*, which eventually will create simulation contexts for all agents that could actually contribute to answering the question whether they believe that they are mortal. The rule handler function also makes sure, that for every reporting agent the instantiated request is issued only once in its simulation context. Since reports to *VarAgP9* are always intercepted and redirected to other contexts, *VarAgP9* remains empty throughout, which is also the reason why it is never shown as a graphical reasoning context. Here are the relevant portions of the SNeSPLOG trace that exemplify this procedure:

```
I wonder if B(a, all(x)(Human(x) => Mortal(x)))
holds in the top-level context

I wonder if Phil(a)
holds in the top-level context

I know Phil(Plato)

Since all(a)(Phil(a) => B(a, all(x)(Human(x) => Mortal(x))))
and Phil(Plato)
I infer B(Plato, all(x)(Human(x) => Mortal(x)))
                                                   ...completed step 6
I know Phil(Soc)

Since all(a)(Phil(a) => B(a, all(x)(Human(x) => Mortal(x))))
and Phil(Soc)
I infer B(Soc, all(x)(Human(x) => Mortal(x)))
                                                   ...completed step 7
                                                   ...opened Plato
                                                   ...opened Soc
```

At this point the report from $\mathsf{B}(\mathsf{Plato}, \forall \mathsf{x}\,(\mathsf{Human}(\mathsf{x}) \Rightarrow \mathsf{Mortal}(\mathsf{x})))$, as well as the one from $\mathsf{B}(\mathsf{Soc}, \forall \mathsf{x}\,(\mathsf{Human}(\mathsf{x}) \Rightarrow \mathsf{Mortal}(\mathsf{x})))$ have accumulated on the incoming reports queue of the node process for $\forall \mathsf{x}\,(\mathsf{Human}(\mathsf{x}) \Rightarrow \mathsf{Mortal}(\mathsf{x})))$. When it finally was the turn of that process to execute again, both reports were handled at once (which explains why the opening operations happened in close succession). Since the reports were directed towards a simulation context with a variable agent, they were intercepted, simulation contexts were opened for Plato and Soc, and requests for Mortal(Plato) and Mortal(Soc) were sent to these new simulation contexts:

```
I wonder if Mortal(Plato)
```

```
holds in the simulation context Plato

I wonder if Mortal(Soc)
holds in the simulation context Soc
```

From here on inference proceeds rather straight-forwardly, since no other relevant agents will report to the `*VarAgP9*` context. Only the trace messages that correspond to reasoning steps have been kept to save space:

```
  ..................

Since B(Plato, all(x)(Human(x) => Mortal(x)))
holds in the top-level context
I will assume that Plato's belief all(x)(Human(x) => Mortal(x))
holds in the simulation context Plato
                                                  ...completed step 8
  ..................

Since B(Soc, all(x)(Human(x) => Mortal(x)))
holds in the top-level context
I will assume that Soc's belief all(x)(Human(x) => Mortal(x))
holds in the simulation context Soc
                                                  ...completed step 9
  ..................

Since all(a)(Phil(a) => B(a, Human(a)))
and Phil(Soc)
I infer B(Soc, Human(Soc))
                                                  ...completed step 10
Since B(Soc, Human(Soc))
holds in the top-level context
I will assume that Soc's belief Human(Soc)
holds in the simulation context Soc
                                                  ...completed step 11
Since all(x)(Human(x) => Mortal(x))
and Human(Soc)
I infer Mortal(Soc)
                                                  ...completed step 12
Since Mortal(Soc)
was derived in the simulation context Soc
under the assumption that Soc believes
    Human(Soc)
and all(x)(Human(x) => Mortal(x))
I infer B(Soc, Mortal(Soc))
                                                  ...completed step 13
Since all(a)(B(a, Mortal(a)) => Smart(a))
and B(Soc, Mortal(Soc))
I infer Smart(Soc)
```

```
                                              ...completed step 14
I know Phil(Plato)

Since all(a)(Phil(a) => B(a, Human(a)))
and Phil(Plato)
I infer B(Plato, Human(Plato))
                                              ...completed step 15
Since B(Plato, Human(Plato))
holds in the top-level context
I will assume that Plato's belief Human(Plato)
holds in the simulation context Plato
                                              ...completed step 16
Since all(x)(Human(x) => Mortal(x))
and Human(Plato)
I infer Mortal(Plato)
                                              ...completed step 17
Since Mortal(Plato)
was derived in the simulation context Plato
under the assumption that Plato believes
    Human(Plato)
and all(x)(Human(x) => Mortal(x))
I infer B(Plato, Mortal(Plato))
                                              ...completed step 18
Since all(a)(B(a, Mortal(a)) => Smart(a))
and B(Plato, Mortal(Plato))
I infer Smart(Plato)
                                              ...completed step 19
 Smart(Soc)
 Smart(Plato)

 CPU time : 11.06
```

At the end we get the desired result, namely, that Cassie believes both Socrates and Plato to be smart. The final state of the reasoning contexts is shown in Figure 6.14. Note, that the alias used in the origin set of the hypothesis in reasoning step 9 is 8, since that is the step number at which it appeared first.

6.8 Oscar and Complexity Theory

The following example shows how SIMBA handles the example about Oscar's understanding of complexity theory discussed in Sections 5.1 and 5.5. It shows how Cassie can simulate Oscar's reasoning, how then a simulation result can become shadowed by a contradicting belief hypothesis, and how belief entailment can be used in such a situation to still represent the fact that Oscar's beliefs entailed the initial simulation result.

The only differences between the previously presented version of this example and the formalization below are that function names were abbreviated to save space, and that a special SNePS

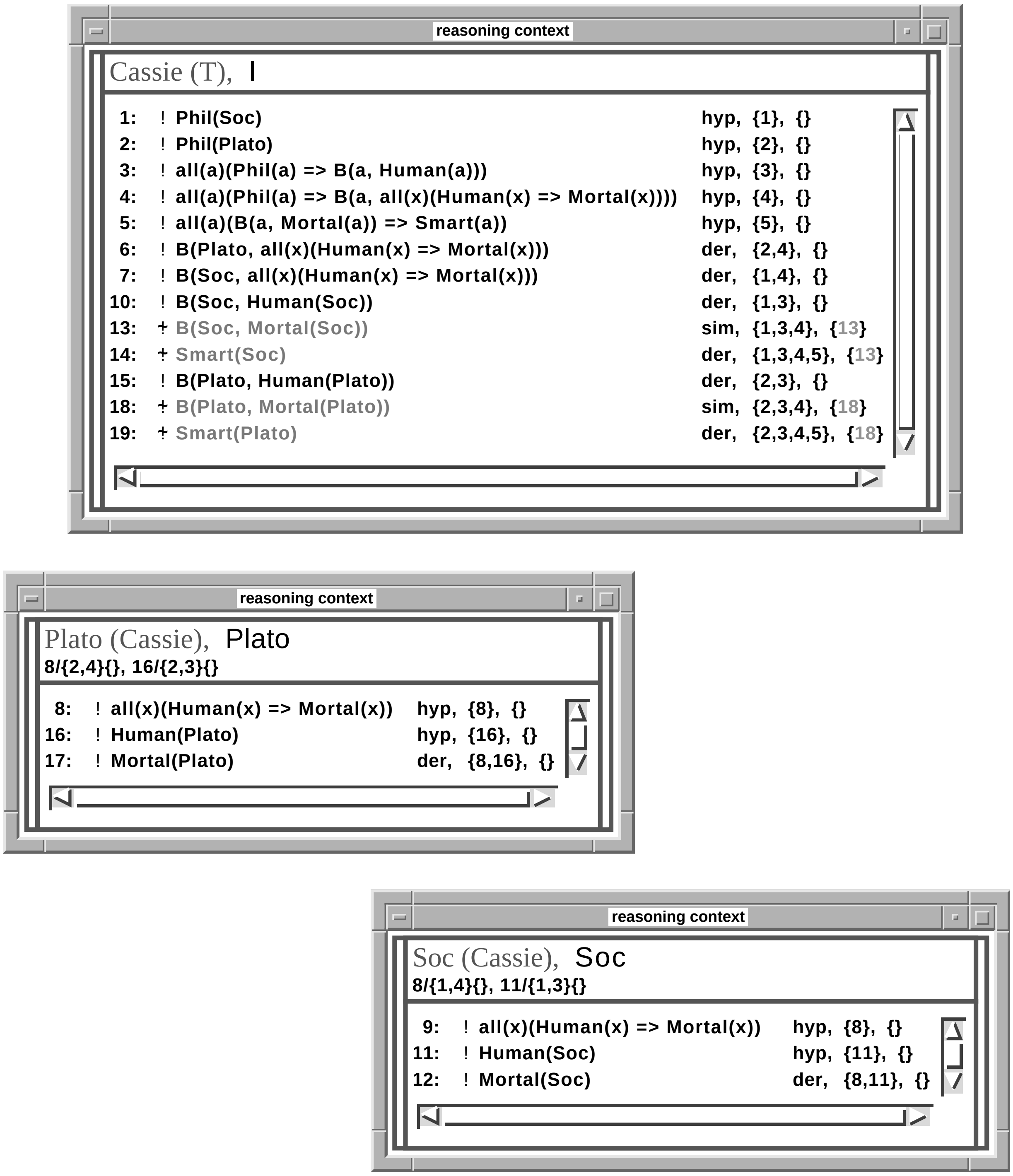


Figure 6.14: Answering a WH-question that involves unspecified agents

connective called an *and-entailment* is used for the first belief. An and-entailment allows one to write a proposition such as

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Rightarrow q$$

with a special set notation for the antecedent that looks like this:

$$\{p_1, p_2, \ldots, p_n\} \wedge\!\!\Rightarrow q$$

The semantics of both notations is the same, i.e., the conjunction of all p_i has to be believed in order to derive the consequent (this is indicated with the special connective $\wedge\!\!\Rightarrow$). However, the implementation can use an and-entailment more efficiently, since it is not necessary to perform extra steps of and-introduction in order to satisfy the antecedent. In SNePSLOG the and-entailment connective is entered as `&=>`. With that in mind, here are Cassie's initial beliefs (an intermediate state of all reasoning contexts is shown in Figure 6.15):

```
: clearkb
                                                    ...opened Cassie
 Knowledge Base Cleared

: B(Oscar, all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)}
:                                       &=> Mem(e, c1))).
                                                    ...completed step 1
 B(Oscar, all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)}
                                        &=> Mem(e, c1)))
: B(Oscar, Cl(P)).
                                                    ...completed step 2
 B(Oscar, Cl(P))

: B(Oscar, Cl(NP)).
                                                    ...completed step 3
 B(Oscar, Cl(NP))

: B(Oscar, all(p)(Mem(p, P) => PolyTime(p))).
                                                    ...completed step 4
 B(Oscar, all(p)(Mem(p, P) => PolyTime(p)))

: B(Oscar, Mem(SAT, NP)).
                                                    ...completed step 5
 B(Oscar, Mem(SAT, NP))
```

Now we are ready to ask Cassie whether Oscar believes the entailment in question. We will not comment on individual inference steps, since those have been explained before. The only thing to keep in mind is that reiteration steps into hypothetical contexts are not annotated explicitly in graphical reasoning contexts. Instead, everything available in the parent context of a hypothetical context will automatically be available in the hypothetical context also. In the inference trace below uninteresting request messages have again been deleted to save space:

: B(Oscar, Eq(P, NP) => PolyTime(SAT))?

```
I wonder if B(Oscar, Eq(P, NP) => PolyTime(SAT))
holds in the top-level context
                                                  ...opened Oscar
I wonder if Eq(P, NP) => PolyTime(SAT)
holds in the simulation context Oscar

Let me assume that Eq(P, NP)
                                                  ...opened Oscar/=>-Intro-1
                                                  ...completed step 6
  ..................

Since B(Oscar, all(p)(Mem(p, P) => PolyTime(p)))
holds in the top-level context
I will assume that Oscar's belief all(p)(Mem(p, P) => PolyTime(p))
holds in the simulation context Oscar
                                                  ...completed step 7
  ..................

Since B(Oscar, all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)}
                                         &=> Mem(e, c1)))
holds in the top-level context
I will assume that Oscar's belief
  all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)} &=> Mem(e, c1))
holds in the simulation context Oscar
                                                  ...completed step 8
  ..................

Since B(Oscar, Cl(P))
holds in the top-level context
I will assume that Oscar's belief Cl(P)
holds in the simulation context Oscar
                                                  ...completed step 9
Since B(Oscar, Cl(NP))
holds in the top-level context
I will assume that Oscar's belief Cl(NP)
holds in the simulation context Oscar
                                                  ...completed step 10
Since B(Oscar, Mem(SAT, NP))
holds in the top-level context
I will assume that Oscar's belief Mem(SAT, NP)
holds in the simulation context Oscar
                                                  ...completed step 11
Since all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)} &=> Mem(e, c1))
and Cl(P)
and Cl(NP)
and Eq(P, NP)
```

```
and Mem(SAT, NP)
I infer Mem(SAT, P)
                                                   ...completed step 12
Since all(p)(Mem(p, P) => PolyTime(p))
and Mem(SAT, P)
I infer PolyTime(SAT)
                                                   ...completed step 13
Since PolyTime(SAT)
was derived assuming Eq(P, NP)
I infer Eq(P, NP) => PolyTime(SAT)
                                                   ...completed step 14
Since Eq(P, NP) => PolyTime(SAT)
was derived in the simulation context Oscar
under the assumption that Oscar believes
    Mem(SAT, NP)
and all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)} &=> Mem(e, c1))
and Cl(P)
and Cl(NP)
and all(p)(Mem(p, P) => PolyTime(p))
I infer B(Oscar, Eq(P, NP) => PolyTime(SAT))
                                                   ...completed step 15
  ..................

 B(Oscar, Eq(P, NP) => PolyTime(SAT))

 CPU time : 6.56
```

The state of the reasoning contexts after Cassie's initial simulation is shown in Figure 6.15. Now imagine that Cassie gave the exam where she found out that Oscar did not believe what she ascribed to him as a result of simulating his reasoning. Hence, she now believes the negation of the simulation result as a belief hypothesis which is modeled with the following assertion (remember, that in SNePSLOG a '~' is used to represent negation):

: **~B(Oscar, Eq(P, NP) => PolyTime(SAT))**.

```
                                                   ...completed step 16
```

This assertion shadows the previously believed simulation result which is displayed in Figure 6.16 (unbelievable propositions are printed in a very light shade of gray). However, Cassie can still derive that Oscar's beliefs do entail the simulation result by using the belief entailment function which is also shown in Figure 6.16:

: **BE(Oscar, Eq(P, NP) => PolyTime(SAT))**?

```
I wonder if BE(Oscar, Eq(P, NP) => PolyTime(SAT))
holds in the top-level context

I know Eq(P, NP) => PolyTime(SAT)
```

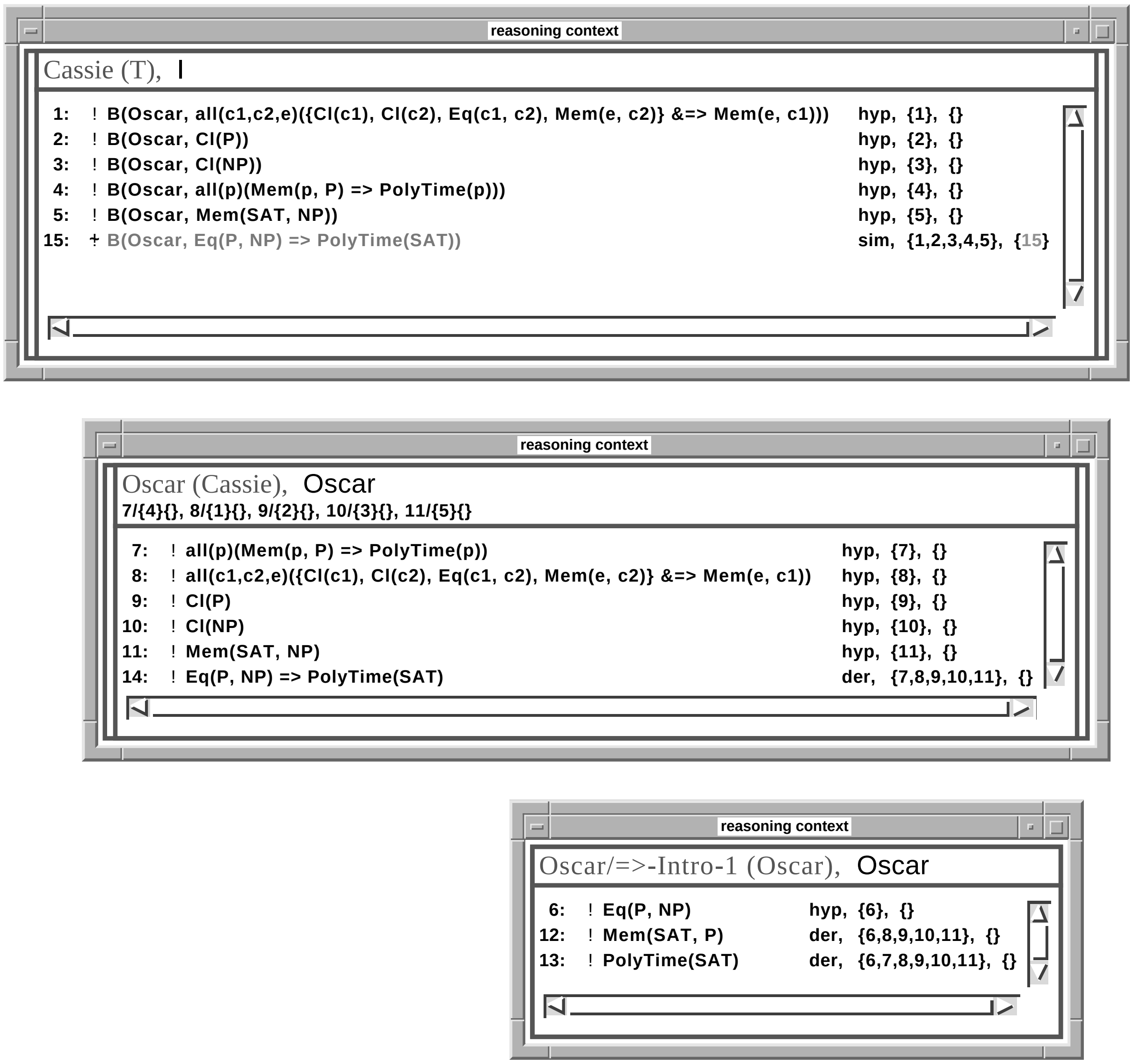


Figure 6.15: Cassie's beliefs about Oscar before the exam

```
Since Eq(P, NP) => PolyTime(SAT)
was derived in the simulation context Oscar
under the assumption that Oscar believes
    Mem(SAT, NP)
and all(c1,c2,e)({Cl(c1), Cl(c2), Eq(c1, c2), Mem(e, c2)} &=> Mem(e, c1))
and Cl(P)
and Cl(NP)
and all(p)(Mem(p, P) => PolyTime(p))
I infer BE(Oscar, Eq(P, NP) => PolyTime(SAT))
                                                    ...completed step 17
 BE(Oscar, Eq(P, NP) => PolyTime(SAT))
```

```
CPU time : 1.18
```

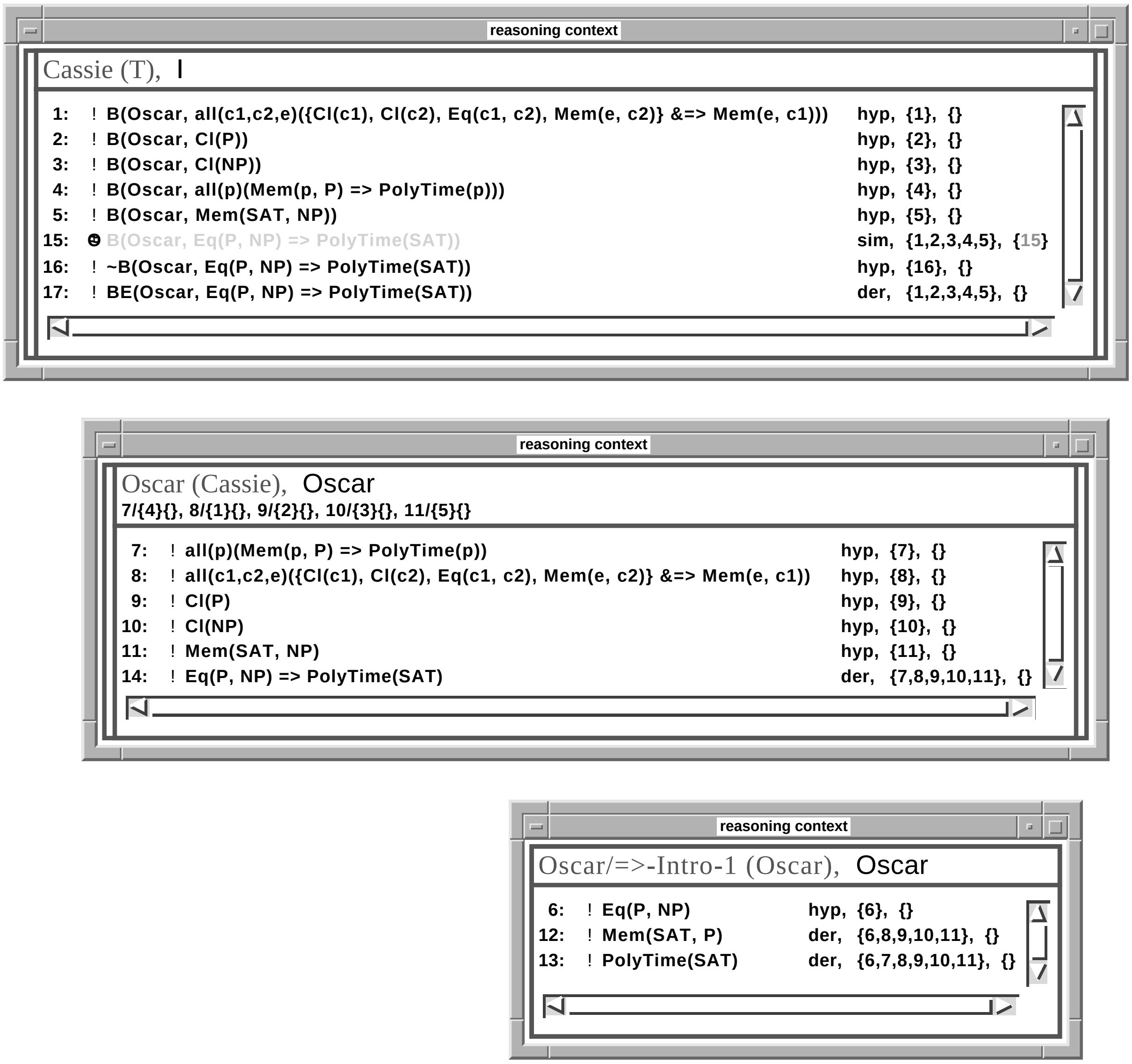


Figure 6.16: Cassie's beliefs about Oscar after the exam

6.9 Multiple Extensions and Their Propagation

This example shows how multiple extensions can ensue, and how they need to be propagated upward and downward according to the definition of extensions given on page 89. The example is somewhat lengthy and admittedly contrived.

Multiple extensions result if a contradiction is based on independent simulation results. In this example this will happen in the following manner: Cassie will simulate John's reasoning in a "middle" John context. John will in turn simulate Lucy's reasoning and derive two simulation results, namely that Lucy believes Q and that she also believes R. These will then allow the derivation of two contradicting propositions S and ¬S, thus, the two simulation assumptions derived by simulating Lucy's reasoning will wind up in separate extensions in the John context. After that John will simulate Mary's reasoning which will be based on the two conflicting simulation assumptions also, thus, the two extensions from the John context will propagate downward into the Mary context and cause multiple extensions there. Here is the set of Cassie's initial beliefs (the final state of the reasoning context is shown in Figure 6.17):

```
: clearkb
                                                    ...opened Cassie
Knowledge Base Cleared

: B(John, B(Lucy, P)).
                                                    ...completed step 1
B(John, B(Lucy, P))

: B(John, B(Lucy, P => Q)).
                                                    ...completed step 2
B(John, B(Lucy, P => Q))

: B(John, B(Lucy, P => R)).
                                                    ...completed step 3
B(John, B(Lucy, P => R))

: B(John, B(Lucy, Q) => S).
                                                    ...completed step 4
B(John, B(Lucy, Q) => S)

: B(John, B(Lucy, R) => ~S).
                                                    ...completed step 5
B(John, B(Lucy, R) => ~S)

: B(John, B(Mary, T)).
                                                    ...completed step 6
B(John, B(Mary, T))

: B(John, S => B(Mary, T => U)).
                                                    ...completed step 7
B(John, S => B(Mary, T => U))

: B(John, (~S) => B(Mary, T => V)).
                                                    ...completed step 8
B(John, ~S => B(Mary, T => V))
```

Now we ask Cassie whether John believes that Mary believes $U \wedge V$. In order to derive this nested belief John (actually, Cassie's simulation of John) will need to derive U and V in the Mary context before the conjunction can be derived there. Incidentally, the derivation of those conjuncts will be based on the two results of John's simulation of Lucy which wound up in separate extensions:

: **B(John, B(Mary, U and V))?**

```
I wonder if B(John, B(Mary, U and V))
holds in the top-level context
                                                    ...opened John
I wonder if B(Mary, U and V)
holds in the simulation context John
                                                    ...opened John/Mary
  ..................

Since B(John, S => B(Mary, T => U))
holds in the top-level context
I will assume that John's belief S => B(Mary, T => U)
holds in the simulation context John
                                                    ...completed step 9
Since B(John, ~S => B(Mary, T => V))
holds in the top-level context
I will assume that John's belief ~S => B(Mary, T => V)
holds in the simulation context John
                                                    ...completed step 10
  ..................

Since B(John, B(Lucy, Q) => S)
holds in the top-level context
I will assume that John's belief B(Lucy, Q) => S
holds in the simulation context John
                                                    ...completed step 11
Since B(John, B(Lucy, R) => ~S)
holds in the top-level context
I will assume that John's belief B(Lucy, R) => ~S
holds in the simulation context John
                                                    ...completed step 12
I wonder if B(Lucy, Q)
holds in the simulation context John
                                                    ...opened John/Lucy
  ..................

Since B(John, B(Lucy, P => Q))
holds in the top-level context
I will assume that John's belief B(Lucy, P => Q)
holds in the simulation context John
                                                    ...completed step 13
Since B(Lucy, P => Q)
holds in the simulation context John
I will assume that Lucy's belief P => Q
holds in the simulation context John/Lucy
                                                    ...completed step 14
Since B(John, B(Lucy, P => R))
```

```
holds in the top-level context
I will assume that John's belief B(Lucy, P => R)
holds in the simulation context John
                                                ...completed step 15
Since B(Lucy, P => R)
holds in the simulation context John
I will assume that Lucy's belief P => R
holds in the simulation context John/Lucy
                                                ...completed step 16
  ...................

Since B(John, B(Lucy, P))
holds in the top-level context
I will assume that John's belief B(Lucy, P)
holds in the simulation context John
                                                ...completed step 17
Since B(Lucy, P)
holds in the simulation context John
I will assume that Lucy's belief P
holds in the simulation context John/Lucy
                                                ...completed step 18
Since P => Q
and P
I infer Q
                                                ...completed step 19
Since P => R
and P
I infer R
                                                ...completed step 20
Since Q
was derived in the simulation context John/Lucy
under the assumption that Lucy believes
    P
and P => Q
I infer B(Lucy, Q)
                                                ...completed step 21
Since R
was derived in the simulation context John/Lucy
under the assumption that Lucy believes
    P => R
and P
I infer B(Lucy, R)
                                                ...completed step 22
Since B(Lucy, Q) => S
and B(Lucy, Q)
I infer S
                                                ...completed step 23
```

Up to this point the method of approximating extensions used by SIMBA had not discovered that the simulation results $\mathsf{B}(\mathsf{Lucy}, \mathsf{Q})$ and $\mathsf{B}(\mathsf{Lucy}, \mathsf{R})$ belong into different extensions. But in the next step $\neg\mathsf{S}$ is derived which contradicts a result previously derived in the John context. Since both S and $\neg\mathsf{S}$ have non-empty assumption supports SIMBA recognizes that the contradiction is based on conflicting assumptions. To resolve the conflict SIMBA takes the union of these assumptions and partitions them into maximal consistent subsets to form extensions. These maximal subsets are the singleton sets $\{\mathsf{B}(\mathsf{Lucy}, \mathsf{Q})\}$ and $\{\mathsf{B}(\mathsf{Lucy}, \mathsf{R})\}$. As a consequence, all sentences derived in the John context that are based on either one of these two sentences will only be possibly believable, since there is at least one extension in which the will not be believed:

```
Since B(Lucy, R) => ~S
and B(Lucy, R)
I infer ~S
                                                  ...completed step 24
I know it is not the case that S

Since S => B(Mary, T => U)
and S
I infer B(Mary, T => U)
                                                  ...completed step 25
Since ~S => B(Mary, T => V)
and ~S
I infer B(Mary, T => V)
                                                  ...completed step 26
```

The next two steps will import simulation hypotheses into the Mary context. But since the belief propositions they are based on have a non-empty assumption support which is not believed in all extensions, $\mathsf{T} \Rightarrow \mathsf{U}$ and $\mathsf{T} \Rightarrow \mathsf{V}$ will become a priori simulation assumptions in the Mary context which wind up in different extensions right from the beginning. Thus the two extensions of the John context now propagated downward to the Mary context. The rationale behind this, as explained before, is that since John cannot believe $\mathsf{B}(\mathsf{Mary}, \mathsf{T} \Rightarrow \mathsf{U})$ and $\mathsf{B}(\mathsf{Mary}, \mathsf{T} \Rightarrow \mathsf{V})$ in the same frame of mind, he should not be able to believe the associated object propositions in the same frame of mind during the simulation of Mary's reasoning:

```
Since B(Mary, T => U)
holds in the simulation context John
I will assume that Mary's belief T => U
holds in the simulation context John/Mary
                                                  ...completed step 27
Since B(Mary, T => V)
holds in the simulation context John
I will assume that Mary's belief T => V
holds in the simulation context John/Mary
                                                  ...completed step 28
  ..................

Since B(John, B(Mary, T))
holds in the top-level context
```

```
I will assume that John's belief B(Mary, T)
holds in the simulation context John
                                                   ...completed step 29
Since B(Mary, T)
holds in the simulation context John
I will assume that Mary's belief T
holds in the simulation context John/Mary
                                                   ...completed step 30
Since T => U
and T
I infer U
                                                   ...completed step 31
Since T => V
and T
I infer V
                                                   ...completed step 32
Since U
and  V
I infer U and V
                                                   ...completed step 33
 CPU time : 16.81
```

Now the simulation of Mary's reasoning is complete and the conjunction we asked for was derived in the Mary context. However, since it is based on two a priori simulation assumptions which are in different extensions, it is unbelievable. For that reason it does not get introduced as a simulation result in the John context, because that result would be unbelievable also. The problematic assumptions in the John context have not yet propagated upward to the Cassie context, because we have not explicitly asked for them. This is what we do now:

: **B**(**John**, **B**(**Mary**, **T** => **U**))?

```
I wonder if B(John, B(Mary, T => U))
holds in the top-level context

I know B(Mary, T => U)

Since B(Mary, T => U)
was derived in the simulation context John
under the assumption that John believes
    B(Lucy, Q) => S
and S => B(Mary, T => U)
and B(Lucy, P)
and B(Lucy, P => Q)
I infer B(John, B(Mary, T => U))
                                                   ...completed step 34
 B(John, B(Mary, T => U))

 CPU time : 1.35
```

: **B**(**John**, **B**(**Mary**, **T** $=>$ **V**))?

```
I wonder if B(John, B(Mary, T => V))
holds in the top-level context

I know B(Mary, T => V)

Since B(Mary, T => V)
was derived in the simulation context John
under the assumption that John believes
    B(Lucy, P => R)
and B(Lucy, R) => ~S
and B(Lucy, P)
and ~S => B(Mary, T => V)
I infer B(John, B(Mary, T => V))
                                                    ...completed step 35
 B(John, B(Mary, T => V))

 CPU time : 2.00
```

These last two steps show how the multiple extensions of the John context propagated upward to the Cassie context once we introduced simulation results there that were based on them. The final state of all reasoning contexts is shown in Figure 6.17.

6.10 Changing the Believability of Plausibility Assumptions

This example demonstrates how the believability of plausibility assumption changes once their source proposition is not plausibly believable anymore because of multiple extensions. First, let us start out by deriving a plausibility assumption based on a very simple simulation (the associated graphical reasoning contexts are shown in Figure 6.18):

: **clearkb**

```
                                                    ...opened Cassie
 Knowledge Base Cleared
```

: **B**(**Lucy**, **P**).

```
                                                    ...completed step 1
 B(Lucy, P)
```

: **B**(**Lucy**, **P** $=>$ **Q**).

```
                                                    ...completed step 2
 B(Lucy, P => Q)
```

: **Plaus**(**B**(**Lucy**, **Q**))?

```
I wonder if Plaus(B(Lucy, Q))
```

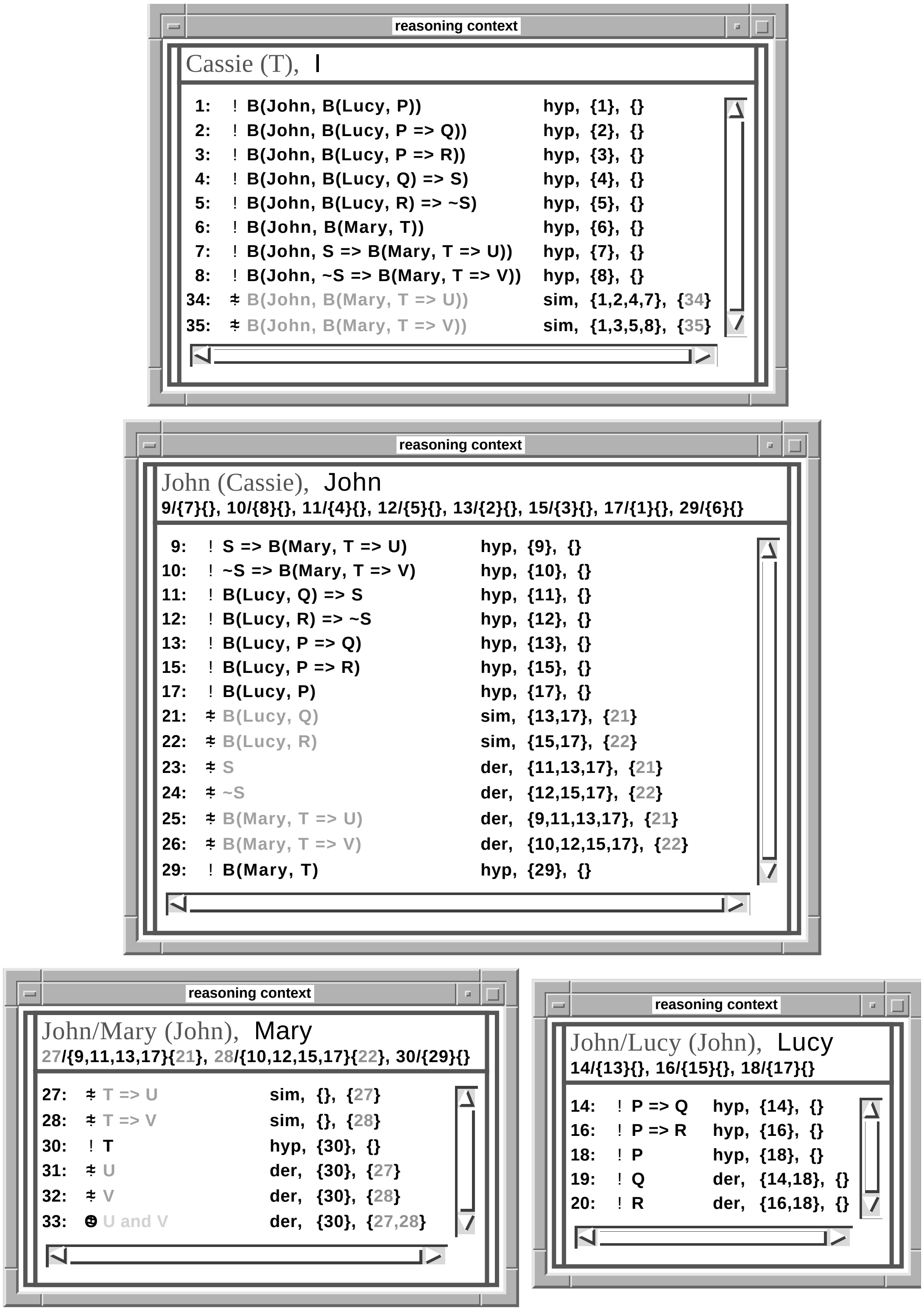


Figure 6.17: Multiple extensions propagate downward and upward

```
holds in the top-level context
                                                  ...opened Lucy
  ...................

Since B(Lucy, P => Q)
holds in the top-level context
I will assume that Lucy's belief P => Q
holds in the simulation context Lucy
                                                  ...completed step 3
I wonder if P
holds in the simulation context Lucy

Since B(Lucy, P)
holds in the top-level context
I will assume that Lucy's belief P
holds in the simulation context Lucy
                                                  ...completed step 4
Since P => Q
and P
I infer Q
                                                  ...completed step 5
Since Q
was derived in the simulation context Lucy
under the assumption that Lucy believes
    P
and P => Q
I infer B(Lucy, Q)
                                                  ...completed step 6
Since B(Lucy, Q)
is at least plausible in the top-level context
I infer Plaus(B(Lucy, Q))
                                                  ...completed step 7
 Plaus(B(Lucy, Q))

 CPU time : 2.21
```

Now we give Cassie some additional beliefs that will lead to multiple extensions similar to the example described in the previous section:

: **B(Lucy, P => R1).**

```
                                                  ...completed step 8
 B(Lucy, P => R1)
```

: **B(Lucy, P => R2).**

```
                                                  ...completed step 9
 B(Lucy, P => R2)
```

: **Plaus(B(Lucy, R1)) => S1.**

```
                                                ...completed step 10
 Plaus(B(Lucy, R1)) => S1
```

: **Plaus(B(Lucy, R2)) => S2.**

```
                                                ...completed step 11
 Plaus(B(Lucy, R2)) => S2
```

: **B(Lucy, R1) => T.**

```
                                                ...completed step 12
 B(Lucy, R1) => T
```

: **B(Lucy, R2) => ~T.**

```
                                                ...completed step 13
 B(Lucy, R2) => ~T
```

As done before we will force a contradiction based on simulation assumptions which will lead to multiple extensions. First, however, we will ask for $\mathsf{S1} \wedge \mathsf{S2}$ which will trigger necessary simulations of Lucy's reasoning and derive various plausibility assumptions on the way:

: **S1 and S2?**

```
I wonder if S1 and S2
holds in the top-level context

  ..................

Since B(Lucy, P => R1)
holds in the top-level context
I will assume that Lucy's belief P => R1
holds in the simulation context Lucy
                                                ...completed step 14
Since B(Lucy, P => R2)
holds in the top-level context
I will assume that Lucy's belief P => R2
holds in the simulation context Lucy
                                                ...completed step 15
I know P

Since P => R2
and P
I infer R2
                                                ...completed step 16
Since P => R1
and P
I infer R1
                                                ...completed step 17
Since R2
was derived in the simulation context Lucy
```

```
under the assumption that Lucy believes
    P => R2
and P
I infer B(Lucy, R2)
                                                  ...completed step 18
Since R1
was derived in the simulation context Lucy
under the assumption that Lucy believes
    P => R1
and P
I infer B(Lucy, R1)
                                                  ...completed step 19
Since B(Lucy, R2)
is at least plausible in the top-level context
I infer Plaus(B(Lucy, R2))
                                                  ...completed step 20
Since B(Lucy, R1)
is at least plausible in the top-level context
I infer Plaus(B(Lucy, R1))
                                                  ...completed step 21
Since Plaus(B(Lucy, R2)) => S2
and Plaus(B(Lucy, R2))
I infer S2
                                                  ...completed step 22
Since Plaus(B(Lucy, R1)) => S1
and Plaus(B(Lucy, R1))
I infer S1
                                                  ...completed step 23
Since S1
and  S2
I infer S1 and S2
                                                  ...completed step 24
 S1 and S2

 CPU time : 6.81
```

What the various reasoning contexts look like at this point is shown in Figure 6.18. Note, that all plausibility assumptions of the form $\mathsf{Plaus}(p)$ are plausibly believable at this point in time, since, again, our method of approximating extensions has not discovered any conflicts yet. Now we will ask for T which will also derive $\neg\mathsf{T}$ as a side-effect:

: **T?**

```
I wonder if T
holds in the top-level context

I wonder if ~T
holds in the top-level context
```

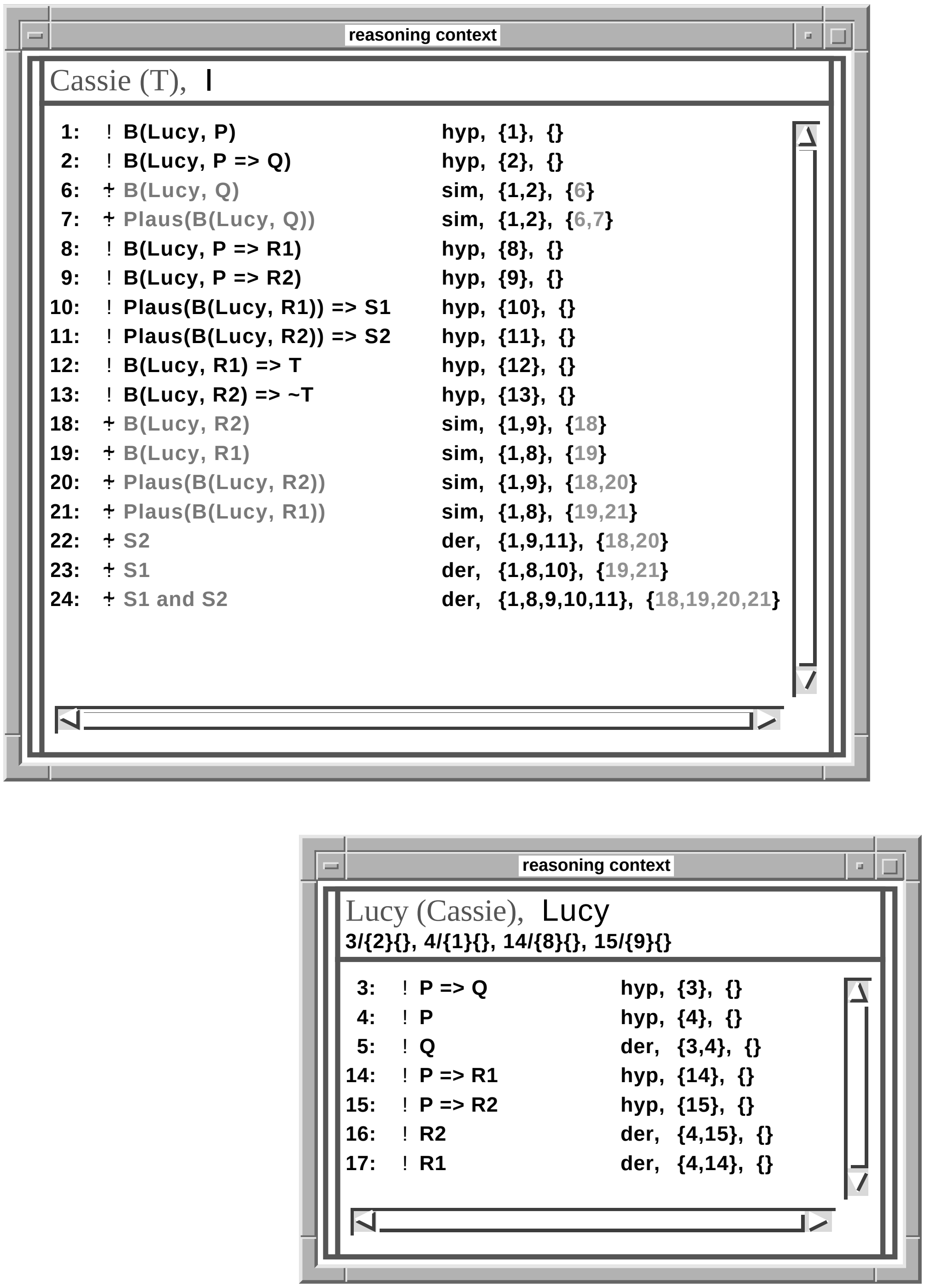


Figure 6.18: Deriving plausibility assumptions

```
I know B(Lucy, R1)

Since B(Lucy, R1) => T
and B(Lucy, R1)
I infer T
                                                    ...completed step 25
I know B(Lucy, R2)

Since B(Lucy, R2) => ~T
and B(Lucy, R2)
I infer ~T
                                                    ...completed step 26
I know it is not the case that T

 T
 ~T

 CPU time : 2.61
```

Once both T and $\neg\mathsf{T}$ are derived in the top-level context, SIMBA creates multiple extensions for the assumptions on which the contradiction is based. At that point some of the plausibility assumptions become automatically unbelievable, since their source propositions are not believed in all extensions anymore. What the reasoning contexts look like after this believability change is shown in Figure 6.19.

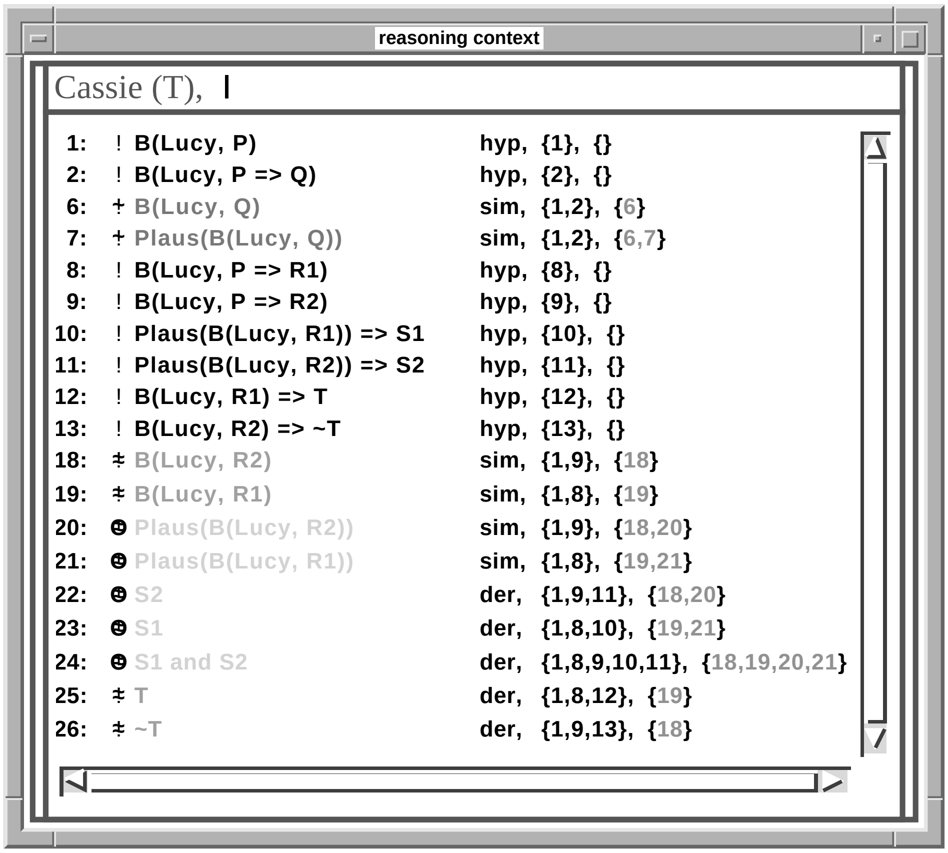
6.11 The Wise Man Puzzle

The Wise Man puzzle is a famous benchmark problem for formalisms that deal with the formalization of knowledge and belief. Its traditional form is due to McCarthy [1979]. Here is a slightly revised version taken from [Konolige, 1986]:

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

In the formalization given below, we let Cassie play the role of the wisest "wise person". She will refer to herself as I, and to the other two wise men as A and B. To express that wise man A has a white spot on his forehead we will write $\mathsf{W(A)}$, to express that his spot is black we will write $\neg\mathsf{W(A)}$. To solve the puzzle Cassie has to represent the following:

1. That at least one of them has a white spot, and that the other two also have a belief to that effect as well as believe it about each other. For example, Cassie's belief that A believes that B believes that at least one of them has a white spot can be represented as the sentence $!\mathsf{B(A, B(B, W(I) \vee W(A) \vee W(B)))}$.



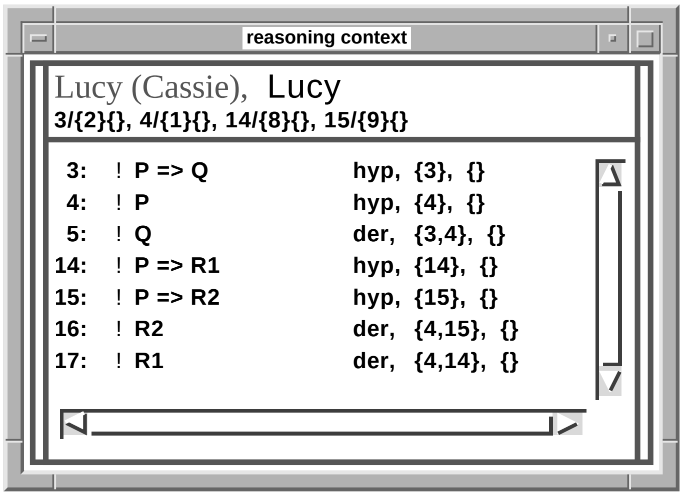


Figure 6.19: The believability of plausibility assumptions can change

2. Facts about mutual observability, for example, if Cassie's spot is of a certain color then the other two wise men can see that and, hence, will have beliefs to that effect. A sentence such as $!\neg \mathsf{W}(\mathsf{I}) \Rightarrow \mathsf{B}(\mathsf{A}, \mathsf{B}(\mathsf{B}, \neg \mathsf{W}(\mathsf{I})))$ can be used to represent Cassie's belief that "if my spot is black then A believes that B believes that it is black."

3. Facts about the reasoning of the two other wise men. For example, since after a while neither of the two has announced the color of his spot, Cassie can assume that they neither believe that their spot is white, nor that their spot is black. In some versions of the puzzle this inference is supported further by including another sentence into the formulation of the puzzle, where first one wise man and then the other announces that he does not know the color of his spot.

The current implementation of SIMBA can only solve a simplified version of the puzzle that involves two wise men. For that reason, we will first give a solution of the puzzle in **SL** to show that the logic is powerful enough to describe and solve the unrestricted version, and then follow that with an example run of SIMBA that solves a simplified version.

6.11.1 A Solution to the Puzzle in SL

The **SL**-solution to the puzzle is shown in Figure 6.20. To keep it as simple and short as possible, we only represent the bare minimum of beliefs necessary to solve the puzzle. For example, the sentence of step 1 represents Cassie's belief that if her spot is black then A believes that B believes that it is black. This is a very specific instance of a more general rule about what is observable by the two other wise men based on the particular situation described in the puzzle. The rule can be paraphrased as: "If my spot is black, then A must believe that it is black (since he can see me), and B must believe that it is black (since he can also see me), and A must believe that B believes that it is black (since A knows that B can see me), and B must believe that A believes that it is black (since B knows that A can see me), etc." A more general way of representing this would be the following which uses the notion of *common belief* described in Section 2.9 (to make this realistic, we would need to define a more specific notion of common belief that would only apply to the group of wise men and not to arbitrary agents):

$$!\neg \mathsf{W}(\mathsf{I}) \Rightarrow \mathsf{CB}(\neg \mathsf{W}(\mathsf{I}))$$

However, the extra inference steps necessary to derive the sentence of step 1 from the common belief representation above would obfuscate the main line of reasoning. For that reason, we make it available right away as a hypothesis.

The sentence of step 2 represents Cassie's belief that A believes that if his spot is black then B also believes that it is black. Thus, it is a variation of **1** but now expressed as a belief of A. This sentence basically says that Cassie believes about A that he can reason about the observations of other wise men just the way she does herself. Again, the following representation using common belief would be more general, and, on top of that, sufficient to derive both sentences **1** and 2, but for simplicity we will supply 2 as a hypothesis:

$$!\mathsf{CB}(\forall \mathsf{a}\, \neg \mathsf{W}(\mathsf{a}) \Rightarrow \mathsf{CB}(\neg \mathsf{W}(\mathsf{a})))$$

Sentence **3** represents Cassie's belief that A believes that B believes that at least one of them has a white spot. It also is a specific instance of a common belief held by all three wise men.

	Cassie ($\top$), I			
1	$!\neg W(I) \Rightarrow B(A, B(B, \neg W(I))),$	hyp,	$\{1\}, \{\}$	H
2	$!B(A, \neg W(A) \Rightarrow B(B, \neg W(A))),$	hyp,	$\{2\}, \{\}$	H
3	$!B(A, B(B, W(I) \vee W(A) \vee W(B))),$	hyp,	$\{3\}, \{\}$	H
4	$!\neg BE(A, W(A)),$	hyp,	$\{4\}, \{\}$	H
5	$!B(A, \neg BE(B, W(B))),$	hyp,	$\{5\}, \{\}$	H
				open Hyp
27	$!W(I),$	der,	$\{1, 2, 3, 4, 5\}, \{\}$	$\Rightarrow$E 26

	Hyp (Cassie), I			
6	$!\neg W(I),$	hyp,	$\{6\}, \{\}$	H
7	$!B(A, B(B, \neg W(I))),$	der,	$\{1, 6\}, \{\}$	$\Rightarrow$E 1,6
				open A
23	$!BE(A, W(A)),$	der,	$\{1, 2, 3, 5, 6\}, \{\}$	BEI 22
24	$!BE(A, W(A)) \wedge \neg BE(A, W(A),$	der,	$\{1, 2, 3, 4, 5, 6\}, \{\}$	$\wedge$I 4,23
25	$!\neg\neg W(I),$	der,	$\{1, 2, 3, 4, 5\}, \{\}$	$\neg$I 24
26	$!W(I),$	der,	$\{1, 2, 3, 4, 5\}, \{\}$	$\neg\neg$E 25

	A (Hyp), A, 8/$\{1, 6\}\{\}$,9/$\{2\}\{\}$,10/$\{3\}\{\}$,12/$\{5\}\{\}$			
8	$!B(B, \neg W(I)),$	hyp,	$\{8\}, \{\}$	SH 7
9	$!\neg W(A) \Rightarrow B(B, \neg W(A)),$	hyp,	$\{9\}, \{\}$	SH 2
10	$!B(B, W(I) \vee W(A) \vee W(B)),$	hyp,	$\{10\}, \{\}$	SH 3
11	$!\neg BE(B, W(B)),$	hyp,	$\{11\}, \{\}$	SH 5
				open HypA
22	$!W(A),$	der,	$\{8, 9, 10, 11\}, \{\}$	$\Rightarrow$E 21

	HypA (A), A			
12	$!\neg W(A),$	hyp,	$\{12\}, \{\}$	H
13	$!B(B, \neg W(A)),$	der,	$\{9, 12\}, \{\}$	$\Rightarrow$E 9,12
				open B
18	$!BE(B, W(B)),$	der,	$\{8, 9, 10, 12\}, \{\}$	BEI 17
19	$!BE(B, W(B)) \wedge \neg BE(B, W(B),$	der,	$\{8, 9, 10, 11, 12\}, \{\}$	$\wedge$I 11,18
20	$!\neg\neg W(A),$	der,	$\{8, 9, 10, 11\}, \{\}$	$\neg$I 19
21	$!W(A),$	der,	$\{8, 9, 10, 11\}, \{\}$	$\neg\neg$E 20

	B (HypA), B, 14/$\{8\}\{\}$,15/$\{9, 12\}\{\}$,16/$\{10\}\{\}$			
14	$!\neg W(I),$	hyp,	$\{14\}, \{\}$	SH 8
15	$!\neg W(A),$	hyp,	$\{15\}, \{\}$	SH 13
16	$!W(I) \vee W(A) \vee W(B),$	hyp,	$\{16\}, \{\}$	SH 10
17	$!W(B),$	der,	$\{14, 15, 16\}, \{\}$	Thm. 14,15,16

Figure 6.20: A solution to the Wise Man puzzle in **SL**

The crucial observation that enables the wisest wise man (or Cassie) to determine the color of his own spot is that the other two wise men were not able to determine the color of their spot. From that it follows, for example, that wise man A does not believe that his spot is white, since otherwise he would have said so. This non-belief could be represented as $\neg\mathsf{B}(\mathsf{A},\mathsf{W}(\mathsf{A}))$. As will become clear below, however, this is not strong enough, since the core of the solution to the puzzle is a reductio argument which can be paraphrased in the following way: "Suppose my spot is black. Then (according to some further reasoning) A must believe his spot is white, but I know he does not believe that. Contradiction. Hence, my spot must be white." The problematic part is the one that leads to the conclusion that "A must believe his spot is white", since it is based on a simulation of A's reasoning, and a belief derived as the result of a simulation is an assumption which can only lead to an *apparent* contradiction. For the reductio argument, however, we need a *real* contradiction in order to be able to apply the inference rule of negation introduction. The solution to the problem is to reason about belief *entailment* instead of plain belief. In a way, this was already implicit in the phrase "must believe" used above. Thus, in step 4 we represent Cassie's belief that A's beliefs do not *entail* that his spot is white, which is stronger than simply saying that he does not believe that his spot is white. Since wise men can be viewed as somewhat ideal reasoners, it is justified to assume that they do believe whatever their beliefs entail.

Alternatively, the content of the last sentence could be represented directly as follows:

$$!\forall \mathsf{a},\mathsf{p}\,(\mathsf{WiseMan}(\mathsf{a}) \wedge \mathsf{BE}(\mathsf{a},\mathsf{p}) \Rightarrow \mathsf{B}(\mathsf{a},\mathsf{p}))$$

This would be a more elegant way to express that wise men are ideal reasoners, and it would actually suffice to assert $!\neg\mathsf{B}(\mathsf{A},\mathsf{W}(\mathsf{A}))$ in step 4 instead of using the belief entailment, since with the help of the rule given above we could derive belief sentences as hard conclusions instead of simulation assumptions which would then give us the real contradictions necessary for the reductio argument. [3] For simplicity, we will stick with our somewhat more unrealistic representation, since it requires less reasoning steps. Regardless of the particular details of the representation, however, the necessity to use belief entailment for the solution of the Wise Man puzzle further supports the validity of the distinction made by the logic **SL**.

Step 5 represents something similar to step 4, but now as one of A's beliefs about the reasoning of B. Since the description of the puzzle given above does not make it completely obvious why A should have come to this conclusion about B, some versions of the puzzle add another sentence such as "first B and then A announce that they do not know the color of their spot", to give more support to this belief of A.

Let us now go through Cassie's reasoning step by step: In step 6 she hypothetically assumes her spot to be black. Hence, in step 7, she can infer that in this case A believes that B believes that her spot is black (to make the proof shorter we omit reiteration steps and use premises for inference rules directly from parent contexts whenever possible). Now Cassie simulates A's reasoning. Steps 8 to 11 simply import the various propositions believed by A (according to Cassie) as simulation hypotheses into A's simulation context. Cassie's simulation of A very much mirrors her own reasoning. In step 12 he hypothetically assumes the color of his spot to be black. Thus, in step 13 he can derive that in this case B believes that A's spot is black. Now A starts a simulation of B. Steps 14 to 16 import the various propositions believed by B into B's simulation context. In step 17 Cassie's simulation of A's simulation of B leads to the conclusion $!\mathsf{W}(\mathsf{B})$. This last step was really more than one, since there is no individual inference rule that would justify it. However, it can easily be shown as a theorem that whenever we have a disjunction and the negations of all but one of the disjuncts

[3] I owe this observation to John Barnden.

we can infer the remaining disjunct as a conclusion. In step 18 A infers that B's beliefs entail the color of his spot to be white. Using belief entailment introduction here instead of plain belief introduction yields a hard conclusion instead of a simulation assumption, hence, the contradiction derived in step 19 is a real one. Here we are at the heart of the reductio argument described above. Wise man A assumed his spot to be black which led to the conclusion that then B's beliefs entail his spot to be white. But B did not know the color of his spot, hence, contradiction. Since this is a real contradiction, A can use negation introduction to conclude the negation of one of the hypotheses on which the contradiction is based. Choosing to negate the hypothesis introduced in step 12 leads to the derivation of !W(A) in step 21. Step 22 is an application of implication introduction, however, since the origin set of !W(A) does not contain the hypothesis of step 12, the set of antecedents of the introduced implication is empty, thus it is not really an implication at all but simply a copy of the conclusion reached in step 21. Now Cassie is done with the simulation of A's reasoning, and she can conclude in step 23 that A's beliefs entail his spot to be white. With a set of steps similar to steps 19 to 22 she arrives at the conclusion that the color of her spot must be white, which solves the puzzle and wins her the king's favor.

6.11.2 An Example Run Solving a Simplified Version of the Puzzle

The main reason why we cannot solve the full Wise Man puzzle in SIMBA is that its underlying inference engine (SNIP) is not a complete theorem prover. SNIP is geared towards commonsense inference based on chaining of domain rules, and a reductio argument such as the one described above does not fit very well into such a frame work. We can, however, solve a simplified version of the puzzle that involves only two wise men (Cassie and wise man A), by asking the right question which will tease out the contradiction necessary to derive the conclusion as a side effect.

Here is the initial setup of the top-level Cassie context (the final state of the reasoning contexts is shown in Figure 6.21):

```
: clearkb
                                                ...opened Cassie
 Knowledge Base Cleared

: ~W(I) => B(A, ~W(I)).
                                                ...completed step 1
 ~W(I) => B(A, ~W(I))

: B(A, W(I) or W(A)).
                                                ...completed step 2
 B(A, W(I) or W(A))

: ~BE(A, W(A)).
                                                ...completed step 3
 ~BE(A, W(A))
```

Ideally, what we would want to ask Cassie is whether her spot is white. But, for the reasons described above, this will not work. What we can do instead is to lead her on the right way by asking the following: Suppose your spot is black, would that entail that A's beliefs entail that his spot is white? Said differently, we can ask Cassie to prove the entailment $\neg W(I) \Rightarrow BE(A, W(A))$.

This question will lead her to initiate hypothetical reasoning based on the hypothesis $\neg \mathsf{W(I)}$, which in turn will derive the necessary contradiction, and the conclusion as a side effect. Here is how:

: ˜W(I) => BE(A, W(A))?

```
I wonder if ~W(I) => BE(A, W(A))
holds in the top-level context

Let me assume that ~W(I)
                                              ...opened =>-Intro-1
                                              ...completed step 4
I wonder if BE(A, W(A))
holds in the hypothetical context =>-Intro-1
                                              ...opened =>-Intro-1/A
I know it is not the case that BE(A, W(A))

  ...................

Since B(A, W(I) or W(A))
holds in the hypothetical context =>-Intro-1
I will assume that A's belief W(I) or W(A)
holds in the simulation context =>-Intro-1/A
                                              ...completed step 5
  ...................

I know ~W(I)

Since ~W(I) => B(A, ~W(I))
and ~W(I)
I infer B(A, ~W(I))
                                              ...completed step 6
Since B(A, ~W(I))
holds in the hypothetical context =>-Intro-1
I will assume that A's belief ~W(I)
holds in the simulation context =>-Intro-1/A
                                              ...completed step 7
I know it is not the case that W(I)

Since W(I) or W(A)
and it is not the case that W(I)
I infer W(A)
                                              ...completed step 8
Since W(A)
was derived in the simulation context =>-Intro-1/A
under the assumption that A believes
    ~W(I)
and W(I) or W(A)
I infer BE(A, W(A))
```

```
                                                         ...completed step 9
Since BE(A, W(A))
contradicts ~BE(A, W(A))
in the hypothetical context =>-Intro-1
and it was derived assuming ~W(I)
I infer that W(I)
holds in the top-level context

Since BE(A, W(A))
was derived assuming ~W(I)
I infer ~W(I) => BE(A, W(A))
                                                         ...completed step 10
                                                         ...completed step 11
 ~W(I) => BE(A, W(A))

 CPU time : 4.41
```

When the contradiction is detected right after step 9 was completed, the sentence !W(I) is derived as a result. However, since that is a conclusion nobody really asked for, it has to be added with a special report that triggers forward inference. The way the forward-inference report gets scheduled on the queue is responsible for the fact that the actual addition of !W(I) to the top-level context in step 11 is delayed until step 10 has been completed. The result reported at the end of the run is the entailment we asked for, however, as a side effect we also obtained the sentence we were really interested in.

The final state of all reasoning contexts involved in this example is shown in Figure 6.21. Note that the step number of the step on which a particular sentence appears first is used as its alias. This is the reason why within the simulation context for wise man A the alias **4** is used instead of 7 for !¬W(I).

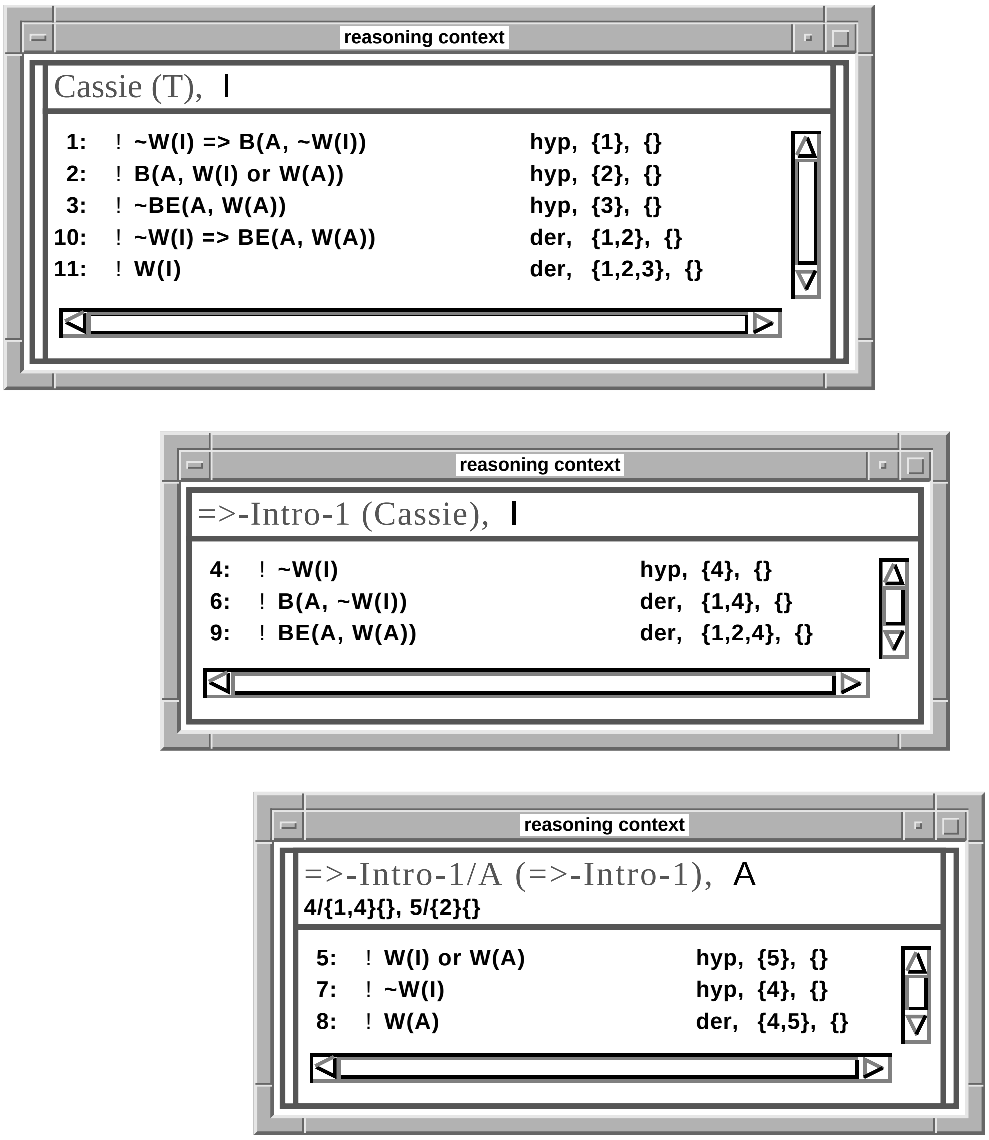


Figure 6.21: Solving a simplified version of the Wise Man puzzle

Chapter 7

Conclusion

Our main result is the development of SIMBA. SIMBA is an implemented belief reasoning system that is intended to be the central building block of "the mind" of an artificial cognitive agent such as Cassie. SIMBA allows Cassie to represent her own beliefs and reason with them similar to what is done in various existing AI systems. The most important new feature of SIMBA, however, is to give Cassie the ability to reason with beliefs she believes are held by other agents. The principle method used to achieve this is *simulative reasoning*, a strategy in which an agent such as Cassie attributes her own reasoning skills to another agent in order to simulate its reasoning. The result of such a simulation is then *ascribed* to the simulated agents as one of its actual beliefs. Since an ascription might not be accurate, for example, the simulated agent might not have done the necessary reasoning itself to hold the ascribed belief, it has to be done *defeasibly*.

The foundation of SIMBA is a fully intensional, subjective, non-monotonic belief logic called **SL**. The language of **SL** serves as a belief representation language or Cassie's language of thought, while the deductive system of the logic models Cassie's reasoning. The belief model underlying **SL** is strongly motivated by the belief model of SNePS, the Semantic Network Processing System. Because of that, it was natural to implement SIMBA as an extension of a previously existing SNePS implementation.

7.1 Main Contributions

Here is a summary of what we consider to be the main contributions of this dissertation:

(1) The Development of the belief logic **SL**, which combines the following features:

Propositional belief model: In **SL**, belief is modeled as a propositional attitude, i.e., as a relation between an agent and a proposition. What sets **SL** apart from other logics based on a propositional attitude model of belief is that in **SL** propositions are treated as actual objects in the domain that can be represented and discussed. In fact, the main representational vehicle in **SL** are proposition-valued functions instead of the commonly used truth-valued predicates. This allows for a particularly simple and natural representation of belief and nested belief, and qualifies the language of **SL** as a proper belief representation language suitable to be used as Cassie's language of thought. **SL** does not provide a definition of what propositions *are*. They are simply viewed as abstractions of things of psychological nature that can somehow be communicated via natural language sentences.

Intensionality: The domain of discourse whose objects are denoted by expressions of **SL** is a set of *intensional* entities such as propositions, objects of thought, fictional objects, etc., rather than a set of extensional individuals that exist in the real or physical world. Since intensional individuals are individuals in their own right and not defined in terms of sets of extensions, this approach is called *full intensionality.* **SL** does not make any assumption about how such intensional individuals get created in the mind of an agent, or how they can be linked to actual, physical individuals in the "real" world. This task is outside the scope of **SL** and left to some unspecified sensory component.

Subjectivity: **SL** is subjective in the sense that it describes a subjective view of the world of one particular agent. Expressions of **SL** are not seen as being assertions *about* an agent from an outside observer's point of view; rather, they actually *make up* or *create* an agent. These expressions are the very mental substrate of an agent and its only means to represent and reason with beliefs. Thus, the same language that is used to represent the agent's own believed propositions is used to represent propositions within nested belief contexts. Similarly, individuals used in belief contexts come from the same mental individual vocabulary that the agent's own individuals are from. Thus, no technical devices such as standard names or translation maps from one agent's language into another as used by some other belief logics (e.g., [Konolige, 1986]) are necessary. The language of **SL** is Cassie's language of thought, and it is the only language she can think in.

Belief nesting, introspection, and quantification: **SL** supports representation of, and reasoning with, arbitrary levels of belief nesting, introspective belief such as "I believe p", and full quantification. Reasoning about the introspection of other agents is also supported.

Semantics: **SL** has a full, formal specification of the semantics of the expressions of its language. For the monotonic version of its deductive system, it can be shown that the system is sound with respect to the specified semantics. Since **SL** is a descendant of the SNePS representation model, this semantics could also serve as a starting point for a formal specification of the semantics of SNePS networks.

Inference and belief reasoning: **SL** has a powerful deductive system based on natural deduction to formalize Cassie's top-level inference as well as her reasoning with the beliefs held by other agents. The core set of inference rules is similar to that of natural deduction systems for standard first-order predicate calculus. This core set is extended with a special set of inference rules for reasoning with belief. The central belief reasoning mechanism is that of simulative reasoning where new beliefs of an agent are derived by simulating that agent's reasoning in a special simulation context. The design of the underlying context system in conjunction with the availability of a complete set of standard logical inference rules also allows for proper reasoning with hypothetical belief, disjunctive belief, as well as negative belief.

Defeasible ascription of belief: **SL** supports the defeasible ascription of beliefs which were derived with help of simulative reasoning. Simulation results are viewed as assumptions that are ascribed to a simulated agent only *by default.* **SL** combines a monotonic belief logic with a default logic to achieve such default ascription. Ascribed beliefs can then be shadowed (automatically disbelieved) by conflicting belief hypotheses that were acquired directly. Thus **SL** allows Cassie to reason about *incomplete* agents who do not necessarily believe all those consequences of their base beliefs that were derived by Cassie during the simulation of their reasoning.

Belief entailment: **SL** has a special notion of belief entailment which can be used to represent and reason about cases where an agent's beliefs entail a certain conclusion, while at the same time the agent does not believe that conclusion. Belief entailment can also be used to formulate the reasoning of ideal agents such as, for example, the wise men in the Wise Man Puzzle.

Handling of multiple extensions: Simulation results that are ascribed to agents by default are viewed as extensions of the underlying base belief set. **SL** can handle multiple extensions that arise from mutually conflicting simulation results. Individual extensions can be viewed as different frames of mind. Multiple extensions are maintained as maximal consistent subsets of the set of all simulation assumptions. The maximality requirement guarantees that the number of extensions is always as small as possible.

Different degrees of believability: **SL** supports a notion of degree of believability for propositions, which depends on whether the propositions are based on any simulation assumptions at all, and if so whether these assumptions are members of all extensions or not. Special inference rules are available to make these believabilities explicit, and to allow Cassie to expressly reason about them.

Support for belief maintenance: Finally, **SL** has built-in support for belief maintenance and revision, since every sentence has an associated support that records on what hypotheses its derivation was based. Since whether a proposition is believed in a particular situation is determined by whether its support set is a subset of the currently believed belief hypotheses, removing any such hypothesis will automatically "disbelieve" all sentences that had that hypothesis as part of their support. These support sets are not computed solely for the purpose of belief maintenance; they also play an important role in the computation of extensions.

To our knowledge, no other belief logic has such a comprehensive set of features as **SL**. We believe that these features make **SL** particularly well suited to serve as the logical foundation for an artificial cognitive agent such as Cassie.

(2) Our second major contribution is the implementation of **SL**'s features described above in a program called SIMBA. By providing an implementation we can actually experiment with the logic to test whether it is adequate for the representation and reasoning purposes we are interested in. The examples described in the previous section show how SIMBA can solve a variety of interesting belief reasoning problems.

The intractability of computing extensions is handled by *approximating* them; thus, instead of using the notion of consistency in the definition of extensions, which in general is impossible to compute, we use the weaker notion of *not known to be inconsistent*. Then, whenever a new inconsistency is discovered, a new, better approximation of all extensions can be computed.

SIMBA more or less directly implements most of the inference rules specified by the deductive system of **SL** with the following exceptions: Inference rules dealing with introspection, some forms of quantifier elimination and introduction, as well as reasoning by cases have not yet been implemented and are left for future work.

7.2 Future Work

Systems such as **SL** or SIMBA are of course never finished. Here is a list of possible improvements and extensions to their current versions:

- Extend the framework of proposition-valued functions to avoid the unwanted semantic asymmetry for some of the logical connective functions. Currently, the terms $\wedge(\mathsf{P},\mathsf{Q})$ and $\wedge(\mathsf{Q},\mathsf{P})$ denote different propositions, but it would be desirable to make the denotation of conjunctive propositions independent of the order of individual conjuncts. One possibility would be to introduce set-valued arguments similar to the cable set notation used in the SNePS network language.

- Provide a completeness proof for the monotonic part of **SL**.

- Provide a semantics for the full, non-monotonic logic. This could probably be done in a way similar to what is done for standard default logics, for example, with an approach based on preferred models.

- Allow hypotheses to be partially ordered to facilitate choices between competing extensions.

- Add default rules to **SL** to allow Cassie to reason with incomplete information. The way default reasoning is used at the moment is restricted to the ascription phase for simulation assumptions.

- Allow **SL** to deal with inconsistent agents. At the moment the only inconsistencies which are handled gracefully are those which in some way or other are based on simulation assumptions.

- Design an explicit representation and reasoning scheme for contexts so Cassie can have beliefs about them and reason with them.

- Design special-purpose inference rules that deal with common and group belief. Even though group belief can be handled already with special domain rules, it would be handled more efficiently at the level of inference rules.

- Complete the implementation of as yet unimplemented inference rules.

- Extend SIMBA's version of the SNePS Inference Package into a complete proof procedure.

Appendix A

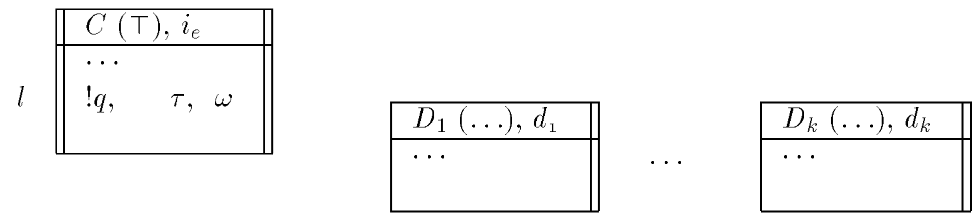
Proofs

Lemma A.1. *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle, n \geq 0$*. Then* $\omega \subseteq !p_1, \ldots, !p_n$*.*

We will not provide a formal proof of this lemma. It basically states that the origin set of a sentence derived in the top-level context of a deduction is always a subset of the set of hypotheses introduced into the context. It can easily be shown that every inference rule generates an origin set that preserves this property. With the lemma available we can prove the following theorem:

Theorem 4.4.1. *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle, n \geq 0$*. Then* $\omega \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega \rangle$

Proof: Since $!p_1, \ldots, !p_n \vdash \langle !q, a/?, \tau, \omega \rangle$, we know that there exists a deduction Δ of the form



with $hyps(C, l) \subseteq \{!p_1, \ldots, !p_n\}$. Suppose $\omega = \{!q_1, \ldots, !q_m\}$. Then Δ can be transformed into the following Δ':

	$C\ (C'),\ i_e$		
	$\ldots$		
l	$!q,$	$\tau,$	$\{!q_1,\ldots,!q_m\}$

$D_1\ (\ldots),\ d_1$
$\ldots$

$\ldots$

	$C'\ (\top),\ i_e$			
$l+1$	$(q_1 \wedge \ldots \wedge q_m) \Rightarrow q,$	`der`,	$\{\}$	$\Rightarrow$I l
$l+2$	$!q_1,$	`hyp`,	$\{!q_1\}$	H
	$\ldots$			
$l+m+1$	$!q_m,$	`hyp`,	$\{!q_m\}$	H
$l+m+2$	$q_{m-1} \wedge q_m,$	`der`,	$\{!q_{m-1}, !q_m\}$	$\wedge$I l+m,l+m+1
	$\ldots$			
$l+2m$	$q_1 \wedge \ldots \wedge q_m,$	`der`,	$\{!q_1,\ldots,!q_m\}$	$\wedge$I l+2,l+2m-1
$l+2m+1$	$q,$	`der`,	$\{!q_1,\ldots,!q_m\}$	$\Rightarrow$E l+1,l+2m

Thus we create an identical copy of the top-level context C of Δ with all its subcontexts, but now we give it the new top-level context C' as its parent. The crucial inference step is step l: By Lemma A.1, $\omega \subseteq hyps(C,l)$, hence, by the definition of $\Rightarrow$I, the implication introduced in step $l+1$ gets ω as its antecedent and an empty origin set. Now we introduce $!q_1,\ldots,!q_m$ one-by-one as hypotheses, and then generate their conjunction in step $l+2m$. Finally, the application of modus ponens derives $!q$ identically to its derivation in Δ. However, in Δ' $hyps(C', l+2m+1) = \omega$ which proves the theorem.

Definition A.1. *Let Δ be a deduction of the following form:*

	$C\ (\top),\ i_e$		
	$\ldots$		
l	$!q,$	$\tau,$	ω

$D_1\ (\ldots),\ d_1$
$\ldots$

$\ldots$

$D_k\ (\ldots),\ d_k$
$\ldots$

Δ *is* **standardized** *if the following conditions hold:*

1. $hyps(C,l) = \omega$
2. *Of all simulation contexts only the top-level context contains any applications of the rule of hypothesis* (H)*.*
3. *Every hypothetical context is used exactly once for either* $\Rightarrow$I, $\forall$I, *or* $\exists$E, *where the characteristic inference step is the last one in the context. Additionally, the following conditions have to hold:*
   (a) $\Rightarrow$I*-contexts contain only one hypothesis, which is introduced in the first step in the context (possibly as a compound hypothesis, i.e., a conjunction).*

(b) $\exists$E*-contexts contain only the relevant hypothesis, and it is introduced in the first step in the context.*

4. *Every application of* $\forall$I *or* $\exists$E *generalizes or eliminates a unique constant that is not used in any other application of* $\forall$I *or* $\exists$E *in* Δ*, and every such constant occurs only in the relevant hypothetical context and possibly its children.*

Standardized deductions restrict the space of possible deductions without restricting the set of derivable sentences. In particular, they eliminate irrelevant applications of the rule of hypothesis in simulation and hypothetical contexts, multi-conclusion hypothetical contexts, and reuse of "i_n-constants" in sibling $\forall$I and $\exists$E-contexts. These restrictions make it easier to perform certain manipulations on deductions and prove certain properties about them. That every deduction can be transformed into a standardized one is expressed by the following lemma:

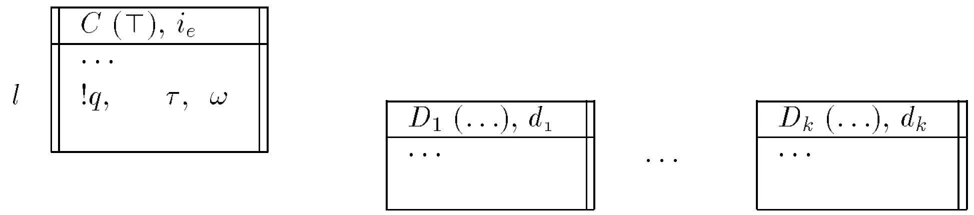
Lemma A.2. *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega\rangle, n \geq 0$*. Then there exists a standardized deduction* Δ *which derives* $\langle !q, a/?, \tau, \omega\rangle$*.*

We will not give a prove of the lemma, since it is rather lengthy but straightforward. It amounts to providing an algorithm with which every deduction can be transformed into a standardized one, and then showing that the resulting deduction still derives the required result.

Theorem 4.5.1. (soundness): *Let* $!p_1, \ldots, !p_n \vdash_{i_e, D_{\mathbf{SL}_0}} \langle !q, a/?, \tau, \omega\rangle, n \geq 0$*, and let a be any consistent agent interpretation structure* $\langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, int\rangle$ *such that* $[\![i_e]\!]_a = \boldsymbol{e}$ *and* $[\![q]\!]_a\!\downarrow$ *(we will call such interpretation structures* **admissible***), and where* $\models_a !p_1, \ldots, \models_a !p_n$*. Then* $\models_a !q$*.*

As a shorthand we will write this as $P \models !q$ with $P = \{!p_1, \ldots, !p_n\}$.

Proof: Since $P \vdash !q$, and because of Lemma A.2, we know that there exists a standardized i_e-deduction Δ for $!q$ of the form



such that $hyps(C, l) = \omega \subseteq P$. We will prove the theorem by induction on the *length* of Δ which is given by the step number l of its conclusion $!q$.

Base case: $l = 1$

Thus $!q$ must be a hypothesis and $\omega = \{!q\} \subseteq P$. Trivially, $P \models !p_i$ for any $!p_i \in P$, hence $P \models !q$.

Induction hypothesis (IH): $l \leq k$

The theorem holds for all $!q$ that are derivable from P in $l \leq k$ steps.

Induction step: $l = k + 1$

Suppose $!q$ is derivable from P in $l = k + 1$ steps. There are 20 cases, one for every inference rule:

a. Δ ended with Rule of Hypothesis (H). Identical to base case.

b. Δ ended with And-Introduction ($\wedge$I). Then $!q$ was of the form $!q_1 \wedge q_2$ and the top-level context looked like this:

$$\begin{array}{r||lll||l}
\hline
 & C\ (\top),\ i_e & & & \\
\hline
 & \dots & & & \\
m & !q_1, & \tau_1, & \omega_1 & \\
 & \dots & & & \\
n & !q_2, & \tau_2, & \omega_2 & \\
 & \dots & & & \\
l & !q_1 \wedge q_2, & \texttt{der}, & \omega_1 \cup \omega_2 & \wedge\mathrm{I}\quad m, n \\
 & & & & \\
\hline
\end{array}$$

$m, n \leq k$, hence, by IH, $P \models\, !q_1$ and $P \models\, !q_2$. Let a be any admissible agent interpretation structure such that $\models_a P$, i.e., $\models_a !p_1, \ldots, \models_a !p_n$. By the above $\models_a !q_1$ and $\models_a !q_2$, hence, by case 3 of Definition 4.3.7, $\models_a !q_1 \wedge q_2$, hence, $\models_a !q$, hence, $P \models\, !q$.

c. Δ ended with And-Elimination ($\wedge$E). Similar to case b.

d. Δ ended with Or-Introduction ($\vee$I). Similar to case b.

e. Δ ended with Or-Elimination ($\vee$E). Then the top-level context looked like this:

$$\begin{array}{r||lll||l}
\hline
 & C\ (\top),\ i_e & & & \\
\hline
 & \dots & & & \\
m & !q_1 \vee q_2, & \tau_1, & \omega_1 & \\
 & \dots & & & \\
n & !q_1 \Rightarrow q, & \tau_2, & \omega_2 & \\
 & \dots & & & \\
o & !q_2 \Rightarrow q, & \tau_3, & \omega_3 & \\
 & \dots & & & \\
l & !q, & \texttt{der}, & \omega_1 \cup \omega_2 \cup \omega_3 & \vee\mathrm{E}\quad m, n, o \\
 & & & & \\
\hline
\end{array}$$

$m, n, o \leq k$, hence, by IH, $P \models\, !q_1 \vee q_2$, and $P \models\, !q_1 \Rightarrow q$, and $P \models\, !q_2 \Rightarrow q$. Let a be admissible and such that $\models_a P$. Suppose $\not\models_a !q$. Since $\models_a !q_1 \Rightarrow q$ it follows by case 5 of Definition 4.3.7 that $\not\models_a !q_1$. Similarly, it follows that $\not\models_a !q_2$. But then by case 4 of Definition 4.3.7 $\not\models_a !q_1 \vee q_2$. Impossible. Hence, $\models_a !q$, and $P \models\, !q$.

f. Δ ended with Double-Negation-Introduction ($\neg\neg$I). Then $!q$ was of the form $!\neg\neg q_1$ and the top-level context looked like this:

$$\begin{array}{r||lll||l}
\hline
 & C\ (\top),\ i_e & & & \\
\hline
 & \dots & & & \\
m & !q_1, & \tau, & \omega & \\
 & \dots & & & \\
l & !\neg\neg q_1, & \texttt{der}, & \omega & \neg\neg\mathrm{I}\quad m \\
 & & & & \\
\hline
\end{array}$$

$m \leq k$, hence, by IH, $P \models !q_1$. Let a be admissible and such that $\models_a P$. Suppose $\models_a !\neg q_1$. But then by case 2 of Definition 4.3.7 $\not\models_a !q_1$. Impossible. Hence, $\not\models_a !\neg q_1$, and, by case 2 of Definition 4.3.7, $\models_a !\neg\neg q_1$, hence, $\models_a !q$, hence, $P \models !q$.

g. Δ ended with Double-Negation-Elimination ($\neg\neg$E). Similar to case f.

h. Δ could not have ended with Negation-Introduction ($\neg$I). Suppose it did, then $!q$ would have been of the form $!\neg q_1$ and the top-level context would have looked like this:

	C ($\top$), i_e			
	...			
m	$!p \wedge \neg p$,	τ,	$\omega \cup \{!q_1\}$	
	...			
l	$!\neg q_1$,	`der`,	$\omega \setminus \{!q_1\}$	$\neg$I m

$m \leq k$, hence, by IH, $P \models !p \wedge \neg p$. Suppose there is an admissible a such that $\models_a P$. Then $\models_a !p \wedge \neg p$. But this is impossible, since an admissible a has to be consistent. Hence, Δ could not have ended with $\neg$I, since in that case there is no admissible agent interpretation structure a such that $\models_a P$.

i. Δ could not have ended with Reiteration (R), since reiteration cannot introduce any sentences into the top-level context. We also do not have to consider Generalized Reiteration (R*), since it is a derived inference rule.

j. Δ ended with Implication-Introduction ($\Rightarrow$I). Since Δ is standardized, $!q$ was of the form $!q_1 \Rightarrow q_2$, and the relevant contexts looked like this:

	C ($\top$), i_e			
	...			
l	$!q_1 \Rightarrow q_2$,	`der`,	$\omega \setminus \{q_1\}$	$\Rightarrow$I m

	D (C), a			
n	$!q_1$,	`hyp`,	$\{q_1\}$	H
	...			
m	$!q_2$,	τ,	ω	

Since Δ is standardized, $hyps(D, m) = \{q_1\}$, and there are no applications of $\Rightarrow$I, $\forall$I, and $\exists$E between steps n and m. Hence we can simply merge the steps of D into the top-level context C to arrive at a deduction $\omega \setminus \{q_1\} \cup \{q_1\} \vdash q_2$ of $\leq m$ steps. Let a be admissible and such that $\models_a P$. Since $\omega \setminus \{q_1\} \subseteq P$, $\models_a \omega \setminus \{q_1\}$. Suppose $\models_a !q_1$. Then, by IH, $\models_a !q_2$, and hence, by case 5 of Definition 4.3.7, $\models_a !q_1 \Rightarrow q_2$. Suppose $\not\models_a !q_1$. Then, by case 5 of Definition 4.3.7, $\models_a !q_1 \Rightarrow q_2$. Therefore, $\models_a !q_1 \Rightarrow q_2$, hence, $\models_a !q$, hence, $P \models !q$.

k. Δ ended with Implication-Elimination ($\Rightarrow$E). Similar to case b.

l. Δ ended with Universal-Introduction ($\forall$I). Then $!q$ was of the form $!\forall x\, q_{1\, in/x}$ and the relevant contexts looked like this:

$$
\begin{array}{l|l|l}
 & C\ (\top),\ i_e & \\
 & \ldots & \\
l & !\forall x\, q_{1\,i_n/x},\ \ \texttt{der},\ \ \omega & \forall\mathrm{I}\ \ m
\end{array}
\qquad
\begin{array}{l|l|}
 & D\ (C),\ a \\
 & \ldots \\
m & !q_1,\ \ \tau,\ \ \omega
\end{array}
$$

To prove this case we need the following simple lemma:

Lemma A.3. *Let Δ be a standardized deduction such that $P \vdash !q$. Let $k_1, \ldots, k_n$ be the individuals affected by applications of $\forall$I and $\exists$E. Let $j_1, \ldots, j_n$ be distinct individuals that do not occur anywhere in Δ. Let Δ' be obtained from Δ by systematically replacing every k_l with j_l. Then Δ' is a standardized deduction for $!q$.*

We will only motivate the lemma but not formally prove it: Since Δ is standardized, all individuals affected by instances of $\forall$I and $\exists$E are unique and only appear in the respective contexts and their children. Hence, it is easy to see that the substitution performed in the lemma conserves that property, and keeps all inference rules used in Δ applicable.

Let a be admissible and such that $\models_a P$. Let $i \in I$ be an arbitrary individual such that $[\![q_{1\,i_n/i}]\!]_a\!\downarrow$. Let $k_1, \ldots k_r$ be the individuals in Δ different from i_n that are affected by applications of $\forall$I and $\exists$E. Let $j_1, \ldots j_r$ be distinct individuals that do not occur in Δ, and that are also different from i. Let Δ' be obtained from Δ by systematically replacing every k_l with j_l. By Lemma A.3 we know that Δ' is a standardized deduction for $!q$ with identical structure.

Now systematically replace i_n with i in Δ', and merge D into the top-level context. Due to the previous global substitution, the introduction of i will not interfere with any applications of $\forall$I and $\exists$E. Since $hyps(D, m) = \emptyset$, in the resulting deduction $P \vdash !q_{1\,i_n/i}$ in $m \leq k$ steps, hence, by IH, $P \models !q_{1\,i_n/i}$, therefore, $\models_a !q_{1\,i_n/i}$ for all such qualifying, arbitrary i. Hence, $\models_a (q_{1\,i_n/x})_{x/i}$ for all such i, hence, by case 6 of Definition 4.3.7, $\models_a !\forall x\, q_{1\,i_n/x}$, hence, $\models_a !q$, hence, $P \models !q$.

m. Δ ended with Universal-Elimination ($\forall$E). Then $!q$ was of the form $!q_{1\,x/i}$ and the top-level context looked like this:

$$
\begin{array}{l|l|l}
 & C\ (\top),\ i_e & \\
 & \ldots & \\
m & !\forall x\, q_1,\ \ \tau,\ \ \omega & \\
 & \ldots & \\
l & !q_{1\,x/i},\ \ \texttt{der},\ \ \omega & \forall\mathrm{E}\ \ m
\end{array}
$$

$m \leq k$, hence, by IH, $P \models !\forall x\, q_1$. Let a be admissible and such that $\models_a P$. Since a is admissible, $[\![q_{1\,x/i}]\!]_a\!\downarrow$, hence, by case 6 of Definition 4.3.7, $\models_a !q_{1\,x/i}$, hence, $\models_a !q$, hence, $P \models !q$.

n. Δ ended with Existential-Introduction ($\exists$I). Then $!q$ was of the form $!\exists x\, q_{1\,i/x}$ and the top-level context looked like this:

$$
\begin{array}{r|l|l}
 & C\ (\top),\ i_e & \\
\hline
 & \cdots & \\
m & !q_1, \quad \tau, \quad \omega & \\
 & \cdots & \\
l & !\exists x\, q_{1\,i/x}, \quad \mathtt{der}, \quad \omega & \exists\mathrm{I}\quad m
\end{array}
$$

$m \le k$, hence, by IH, $P \models !q_1$. Let a be admissible and such that $\models_a P$. Then $\models_a !q_1$, and, since $q_1 = (q_{1x/i})_{i/x}$, $\models_a (q_{1x/i})_{i/x}$. Since a is admissible, and because $P \models !q_1$, $[\![q_1]\!]_a\!\downarrow$ and $[\![(q_{1x/i})_{i/x}]\!]_a\!\downarrow$. Hence, there exists an i such that $[\![(q_{1x/i})_{i/x}]\!]_a\!\downarrow$ and $\models_a (q_{1x/i})_{i/x}$, hence, by case 7 of Definition 4.3.7, $\models_a !\exists x\, q_{1\,i/x}$, hence, $\models_a !q$, hence, $P \models !q$.

o. Δ ended with Existential-Elimination ($\exists$E). Then the relevant contexts looked like this:

$$
\begin{array}{r|l|l}
 & C\ (\top),\ i_e & \\
\hline
 & \cdots & \\
m & !\exists x\, q_1, \quad \tau_1, \quad \omega_1 & \\
 & \cdots & \\
l & !q, \quad \mathtt{der}, \quad \omega_1 \cup \omega_2 & \exists\mathrm{E}\quad m, n, o
\end{array}
\qquad
\begin{array}{r|l|l}
 & D\ (C),\ a & \\
\hline
n & !q_{1x/i_n}, \quad \mathtt{hyp}, \quad \{!q_{1x/i_n}\} & \mathrm{H} \\
 & \cdots & \\
o & !q, \quad \tau_2, \quad \omega_2 &
\end{array}
$$

$m \le k$, hence, by IH, $P \vdash !\exists x\, q_1$. Let a be admissible and such that $\models_a P$. Hence, $\models_a !\exists x\, q_1$, and, by case 7 of Definition 4.3.7, there is some $i \in I$ such that $[\![q_{1x/i}]\!]_a\!\downarrow$ and $\models_a q_{1x/i}$. Let us call it i_0.

Let $k_1, \ldots k_r$ be the individuals in Δ different from i_n that are affected by applications of $\forall$I and $\exists$E. Let $j_1, \ldots j_r$ be distinct individuals that do not occur in Δ, and that are also different from i_0. Let Δ' be obtained from Δ by systematically replacing every k_l with j_l. By Lemma A.3 we know that Δ' is a standardized deduction for $!q$ with identical structure.

Now systematically replace i_n with i_0 in Δ', and merge D into the top-level context. Due to the previous global substitution, the introduction of i_0 will not interfere with any applications of $\forall$I and $\exists$E. Since Δ' is standardized, $hyps(D, m) = \{!(q_{1x/i_n})_{i_n/i_0}\}$, and in the resulting deduction $P \cup \{!(q_{1x/i_n})_{i_n/i_0}\} \vdash !q$ in $o \le k$ steps, hence, by IH, and since $\models_a P$ and $\models_a q_{1x/i_0}$, $\models_a !q$, hence, $P \models !q$.

p. Δ ended with Positive Introspection (+IS). Then $!q$ was of the form $!\mathsf{B}(i_e, q_1)$ and the top-level context looked like this:

$$
\begin{array}{r|l|l}
 & C\ (\top),\ i_e & \\
\hline
 & \cdots & \\
m & !q_1, \quad \tau, \quad \omega & \\
 & \cdots & \\
l & !\mathsf{B}(i_e, q_1), \quad \mathsf{I}, \quad \mathtt{der}, \quad \omega & +\mathrm{IS}\quad m
\end{array}
$$

$m \le k$, hence, by IH, $P \models !q_1$. Let a be admissible and such that $\models_a P$. Then $\models_a !q_1$, and, by case 8 of Definition 4.3.7, $\models_a !\mathsf{B}(i_e, q_1)$, hence, $\models_a !q$, hence, $P \models !q$.

q. Δ ended with Negative Introspection ($-$IS). Then $!q$ was of the form $!\neg\mathsf{B}(i_e, q_1)$ and the top-level context looked like this:

$$
\begin{array}{r|l|l}
\cline{2-2}
 & C\ (\top),\ i_e & \\
\cline{2-2}
 & \cdots & \\
m & !\neg q_1, \qquad \tau, \quad \omega & \\
 & \cdots & \\
l & !\neg\mathsf{B}(i_e, q_1),\ \mathsf{I},\ \mathtt{der},\ \omega & -\text{IS}\ \ m \\
\cline{2-2}
\end{array}
$$

$m \leq k$, hence, by IH, $P \models\, !\neg q_1$. Let a be admissible and such that $\models_a P$, hence, $\models_a\, !\neg q_1$. Suppose $\models_a\, !\mathsf{B}(i_e, q_1)$. But then, by case 8 of Definition 4.3.7, $\models_a\, !q_1$. Impossible, since a is assumed to be admissible and consistent. Hence, $\not\models_a\, !\mathsf{B}(i_e, q_1)$, and, by case 2 of Definition 4.3.7, $\models_a\, !\neg\mathsf{B}(i_e, q_1)$. Hence, $\models_a\, !q$, hence, $P \models\, !q$.

r. Δ ended with Internalization (IN). Similar to case p.

s. Δ could not have ended with Simulation Hypothesis (SH). Similar to case i.

t. Δ ended with Belief Introduction. Then $!q$ was of the form $!\mathsf{B}(b, q_1)$ and the relevant contexts looked like this:

$$
\begin{array}{r|l|l}
\cline{2-2}
 & C\ (\top),\ i_e & \\
\cline{2-2}
 & \cdots & \\
l & !\mathsf{B}(b, q_1),\ \mathtt{der},\ \bigcup_{p_i \in \omega} \omega_i & \mathsf{BI}\ \ m \\
\cline{2-2}
\end{array}
\qquad
\begin{array}{r|l|}
\cline{2-2}
 & D\ (C),\ b, \\
 & p_1/\omega_1, \ldots, p_n/\omega_n \\
\cline{2-2}
 & \cdots \\
m & !q_1, \quad \tau,\ \omega \subseteq \mathit{shyps}(D, m) \\
 & \cdots \\
\cline{2-2}
\end{array}
$$

For every simulation hypothesis $!p_i \in \mathit{shyps}(D, m)$ there exists a corresponding $!\mathsf{B}(b, p_i)$ in C at a step number $\leq m$, hence, by IH, $P \models\, !\mathsf{B}(b, p_i)$. Since Δ is standardized, there are no applications of H in context D up to step m. Hence, we can convert D into a top-level context by introducing all simulation hypotheses as proper hypotheses instead, thus arriving at a deduction $\mathit{shyps}(D, m) \vdash_b\, !q_1$ of $\leq m$ steps. Therefore, by IH, $\mathit{shyps}(D, m) \models !q_1$.

Let $a = \langle \mathcal{D}, \mathcal{F}, \mathcal{B}, \boldsymbol{e}, \mathit{int} \rangle$ be admissible such that $\models_a P$. Suppose $b \neq i_e$: Then $\models_a\, !\mathsf{B}(b, p_i)$ for all $!p_i \in \mathit{shyps}(D, m)$. Let $a' = \langle \mathcal{D}, \mathcal{F}, \{[\![p_1]\!]_a, \ldots, [\![p_n]\!]_a\}, [\![b]\!]_a, \mathit{int} \rangle$. Since a is admissible, it is consistent and, hence, a' is consistent also. Since $\mathcal{D}, \mathcal{F}$, and int are the same for a and a', $[\![b]\!]_{a'} = [\![b]\!]_a$, hence, a' is admissible. Therefore, since $\mathit{shyps}(D, m) \models !q_1$, $\models_{a'}\, !q_1$, hence, by case 9 of Definition 4.3.7, $\models_a\, !\mathsf{B}(b, q_1)$, hence, $\models_a\, !q$, hence, $P \models\, !q$.

Suppose $b = i_e$: Then $\models_a\, !\mathsf{B}(i_e, p_i)$ for all $!p_i \in \mathit{shyps}(D, m)$, hence, by case 8 of Definition 4.3.7, $\models_a\, !p_i$, hence, since $\mathit{shyps}(D, m) \models !q_1$, $\models_a\, !q_1$, hence, by case 8 of Definition 4.3.7, $\models_a\, !\mathsf{B}(i_e, q_1)$, hence, $\models_a\, !q$, hence, $P \models\, !q$. $\square$

Bibliography

[Aiello *et al.*, 1991] L. Aiello, D. Nardi, and M. Schaerf. Reasoning about reasoning in a meta-level architecture. *Journal of Applied Intelligence*, 1:55–67, 1991.

[Anderson and Belnap, 1975] A. R. Anderson and N. D. Belnap. *Entailment: The Logic of Relevance and Necessity.* Princeton University Press, Princeton, NJ, 1975.

[Anderson *et al.*, 1992] A. R. Anderson, N. D. Belnap, and J. M. Dunn. *Entailment: The Logic of Relevance and Necessity*, volume II. Princeton University Press, Princeton, NJ, 1992.

[Appelt, 1985] D. E. Appelt. *Planning English Sentences.* Studies in natural language processing. Cambridge University Press, New York, 1985.

[Arbab, 1988] B. Arbab. A formal language for representation of and reasoning about indirect context. Technical Report CSD-880045, Department of Computer Science, University of California at Los Angeles, Los Angeles, CA, June 1988.

[Arbab, 1989] B. Arbab. How to represent opaque sentences in first order logic. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 458–462, Palo Alto, CA, 1989. Morgan Kaufmann.

[Ballim and Wilks, 1991] A. Ballim and Y. Wilks. *Artificial Believers: The Ascription of Belief.* Erlbaum, Hillsdale, NJ, 1991.

[Ballim *et al.*, 1991] A. Ballim, Y. Wilks, and J. A. Barnden. Belief ascription, metaphor, and intensional identification. *Cognitive Science*, 15(1):133–171, 1991.

[Ballim, 1992] A. Ballim. *ViewFinder: A Framework for Representing, Ascribing and Maintaining Nested Beliefs of Interacting Agents.* PhD thesis, University of Geneva, Geneva, Switzerland, 1992.

[Barnden *et al.*, 1994a] J. A. Barnden, S. Helmreich, E. Iverson, and G. C. Stein. An integrated implementation of simulative, uncertain and metaphorical reasoning about mental states. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference*, San Mateo, CA, 1994. Morgan Kaufmann.

[Barnden *et al.*, 1994b] J. A. Barnden, S. Helmreich, E. Iverson, and G. C. Stein. Combining simulative and metaphor-based reasoning about beliefs. In A. Ram and K. Eiselt, editors, *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 21–26, Hillsdale, NJ, 1994. Lawrence Erlbaum Associates.

[Barnden, 1992] J. A. Barnden. Belief in metaphor: Taking commonsense psychology seriously. *Computational Intelligence*, 8:520–552, 1992.

[Besnard, 1989] P. Besnard. *An Introduction to Default Logic.* Springer-Verlag, Berlin, 1989.

[Bledsoe, 1977] W. W. Bledsoe. Non-resolution theorem proving. *Artificial Intelligence*, 9:1–36, 1977.

[Chalupsky, 1993] Hans Chalupsky. Using hypothetical reasoning as a method for belief ascription. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):119–133, April–September 1993.

[Choi, 1993] J. Choi. *Experienced-Based Learning in Deductive Reasoning Systems.* PhD thesis, Technical Report 93–20, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, May 1993.

[Church, 1950] A. Church. On Carnap's analysis of statements of assertion and belief. *Analysis*, 10:97–99, 1950.

[Cravo and Martins, 1993a] M. R. Cravo and J. P. Martins. Defaults and belief revision: A syntactical approach. Technical Report GIA 93/01, Technical University of Lisbon, Lisbon Portugal: Instituto Superior Técnico, 1993.

[Cravo and Martins, 1993b] M. R. Cravo and J. P. Martins. SNePSwD: A newcomer to the SNePS family. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):135–148, April–September 1993.

[Creary, 1979] L. G. Creary. Propositional attitudes: Fregean representations and simulative reasoning. In *Proceedings of the Sixth International Conference on Artificial Intelligence*, pages 176–181, Palo Alto, CA, 1979. Morgan Kaufmann.

[Fitch, 1952] F. B. Fitch. *Symbolic Logic: An Introduction.* R. Press, New York, 1952.

[Geissler and Konolige, 1986a] C. Geissler and K. Konolige. Implementation of a resolution system for modal logic. Artificial Intelligence Center Tech Note, SRI International, Menlo Park, CA, 1986.

[Geissler and Konolige, 1986b] C. Geissler and K. Konolige. A resolution method for quantified modal logics of knowledge and belief. In J. Y. Halpern, editor, *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 309–324, Palo Alto, CA, 1986. Morgan Kaufmann.

[Grice, 1967] H. P. Grice. Logic and conversation. William James Lectures, Harvard University, unpublished, 1967. Lecture 2 published in P. Cole and J. L. Morgan, editors, *Studies in Syntax*, Vol. III, Seminar Press, New York, 1975.

[Guha, 1991] R. V. Guha. *Contexts: A Formalization and Some Applications.* Technical Report No. STAN-CS-91-1399, Stanford University, 1991.

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

[Haas, 1990] A. R. Haas. Sentential semantics for propositional attitudes. *Computational Linguistics*, 16:213–233, 1990.

[Halpern and Moses, 1992] J. Y. Halpern and Y. O. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, NY, 1962.

[Hull, 1986] R. G. Hull. A new design for SNIP the SNePS inference package. SNeRG Technical Note 14, Department of Computer Science, SUNY at Buffalo, 1986.

[Kaplan, 1968] D. Kaplan. Quantifying in. *Synthèse*, 19:178–214, 1968.

[Konolige, 1982] K. Konolige. A first-order formalization of knowledge and action for a multiagent planning system. In J. E. Hayes, D. Michie, and Y. H. Pao, editors, *Machine Intelligence 10*, pages 120–147. Ellis Horwood Limited, Chichester, England, 1982.

[Konolige, 1986] K. Konolige. *A Deduction Model of Belief*. Morgan Kaufmann, Palo Alto, CA, 1986.

[Kripke, 1963] S. A. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[Kumar, 1994] D. Kumar. *From Beliefs and Goals to Intentions and Actions: An Amalgamated Model of Inference and Acting*. PhD thesis, Technical Report 94–04, Department of Computer Science, State University of New York at Buffalo, 1994.

[Lammens, 1994] J. M. Lammens. *A Computational Model of Color Perception and Color Naming*. PhD thesis, Technical Report 94–26, Department of Computer Science, State University of New York at Buffalo, 1994.

[Levesque, 1982] H. J. Levesque. A formal treatment of incomplete knowledge bases. Technical Report 614, Fairchild, 1982. FLAIR Technical Report No. 3.

[Levesque, 1984] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 198–202, Palo Alto, CA, 1984. Morgan Kaufmann.

[Loveland, 1984] D. W. Loveland. Automated theorem proving: A quarter century review. In W. W. Bledsoe and D. W. Loveland, editors, *Contemporary Mathematics: Automated Theorem Proving - After 25 Years*, pages 1–45. American Mathematical Society, Providence, RI, 1984.

[Maida and Shapiro, 1982] A. S. Maida and S. C. Shapiro. Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330, 1982. Reprinted in R. J. Brachman and H. J. Levesque, eds. *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA, 1985, 170–189.

[Maida, 1983] A. S. Maida. Knowing intensional individuals, and reasoning about knowing intensional individuals. In *Proceedings of the Eighth International Conference on Artificial Intelligence*, pages 382–384, Palo Alto, CA, 1983. Morgan Kaufmann.

[Maida, 1986] A. S. Maida. Introspection and reasoning about the beliefs of other agents. In *Proceedings of the Eights Annual Conference of the Cognitive Science Society*, pages 187–195, Hillsdale, NJ, 1986. Erlbaum.

[Martins and Cravo, 1991] J. P. Martins and M. R. Cravo. How to change your mind. *Noûs*, 25(4):537–551, 1991.

[Martins and Shapiro, 1988] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35(1):25–79, 1988.

[Martins, 1983] J. P. Martins. *Reasoning in Multiple Belief Spaces.* PhD thesis, Technical Report 203, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, 1983.

[Matos and Martins, 1989] P. A. Matos and J. P. Martins. SNePSLOG - a logic interface to SNePS. Technical Report 89/03, Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, 1989.

[McCarthy *et al.*, 1979] J. McCarthy, M. Sato, T. Hayashi, and S. Igarashi. On the model theory of knowledge. Technical Report STAN-CS-78-657, Stanford University, Stanford, CA, 1979.

[McCarthy, 1979] J. McCarthy. First order theories of individual concepts and propositions. In B. Meltzer and D. Michie, editors, *Machine Intelligence 9*, pages 120–147. Ellis Horwood, 1979.

[McCarthy, 1993] J. McCarthy. Notes on formalizing context. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence*, pages 555–560, 1993.

[McDermott and Sussman, 1972] D. McDermott and G. J. Sussman. The Conniver reference manual. AI Lab Memo 259, Massachusetts Institute of Technology, Cambridge, Mass., May 1972.

[McKay and Shapiro, 1980] D. P. McKay and S. C. Shapiro. MULTI — a LISP based multiprocessing system. In *Proceedings of the 1980 LISP Conference*, pages 29–37. Stanford University, Stanford, CA, 1980.

[McKay and Shapiro, 1981] D. P. McKay and S. C. Shapiro. Using active connection graphs for reasoning with recursive rules. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 368–374, Los Altos, CA, 1981. Morgan Kaufmann.

[Montague, 1974] R. Montague. Universal grammar. In R. H. Thomason, editor, *Formal Philosophy: Selected Papers of R. Montague*, chapter 7, pages 222–246. Yale University Press, 1974.

[Moore, 1973] R. C. Moore. D-SCRIPT: A computational theory of descriptions. In *Advance Papers of the Third International Conference on Artificial Intelligence*, pages 223–229, Stanford, CA, 1973. Reprinted in *IEEE Transactions on Computers*, 25(4):366–373, 1976.

[Moore, 1977] R. C. Moore. Reasoning about knowledge and action. In *Proceedings of the Fifth International Conference on Artificial Intelligence*, pages 223–227, Palo Alto, CA, 1977. Morgan Kaufmann.

[Moore, 1980] R. C. Moore. Reasoning about knowledge and action. Technical Note 191, SRI International, Menlo Park, CA, October 1980.

[Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, chapter 9, pages 319–358. Ablex Publishing Corp., Norwood, NJ, 1985.

[Nilsson, 1977] N. Nilsson. A production system for automatic deduction. Computer Science Department Report STAN-CS-77-618, Stanford University, Stanford, CA, July 1977.

[Nutter, 1983] J. T. Nutter. What else is wrong with non-monotonic logics?: Representational and informational shortcomings. In *Proceedings of the Fifth Annual Meeting of the Cognitive Science Society*, page 5, Rochester, NY, 1983.

[Plaisted, 1990] D. A. Plaisted. Mechanical theorem proving. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Sourcebook*, pages 269–320. North-Holland, Amsterdam, 1990.

[Poole *et al.*, 1987] D. Poole, R. Goebel, and R. Aleliunas. Theorist: A logical reasoning system for defaults and diagnosis. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352. Springer–Verlag, New York, 1987.

[Rapaport *et al.*, 1986] W. J. Rapaport, S. C. Shapiro, and J. M. Wiebe. Quasi-indicators, knowledge reports, and discourse. Technical Report 86–15, Department of Computer Science, SUNY at Buffalo, 1986.

[Rapaport, 1986] W. J. Rapaport. Logical foundations for belief representation. *Cognitive Science*, 10:371–422, 1986.

[Rapaport, 1992] W. J. Rapaport. Belief representation systems. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*. Wiley, New York, second edition, 1992.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41, 1965.

[Shapiro and Rapaport, 1987] S. C. Shapiro and W. J. Rapaport. SNePS considered as a fully intensional propositional semantic network. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 263–315. Springer–Verlag, New York, 1987.

[Shapiro and Rapaport, 1991] S. C. Shapiro and W. J. Rapaport. Models and minds: Knowledge representation for natural-language competence. In Robert Cummins and John Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 215–259. MIT Press, Cambridge, MA, 1991.

[Shapiro and Rapaport, 1992] S. C. Shapiro and W. J. Rapaport. The SNePS family. *Computers & Mathematics with Applications*, 23(2–5):243–275, January–March 1992.

[Shapiro and Wand, 1976] S. C. Shapiro and M. Wand. The relevance of relevance. Technical Report 46, Computer Science Department, Indiana University, Bloomington, IN, 1976.

[Shapiro *et al.*, 1981] S. C. Shapiro, D. P. McKay, J. Martins, and E. Morgado. SNePSLOG: A "higher order" logic programming language. SNeRG Technical Note 8, Department of Computer Science, SUNY at Buffalo, 1981. Presented at the Workshop on Logic Programming for Intelligent Systems, R.M.S. Queen Mary, Long Beach, CA.

[Shapiro *et al.*, 1982] S. C. Shapiro, J. Martins, and D. McKay. Bi-directional inference. In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, pages 90–93, Ann Arbor, MI, 1982. the Program in Cognitive Science of The University of Chicago and The University of Michigan.

[Shapiro, 1979] S. C. Shapiro. The SNePS semantic network processing system. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*, pages 179–203. Academic Press, New York, 1979.

[Shapiro, 1991] S. Shapiro. *Foundations without Foundationalism: A Case For Second-Order Logic.* Clarendon Press, Oxford, 1991.

[Shapiro, 1993] S. C. Shapiro. Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):225–235, April–September 1993.

[Smullyan, 1968] R. M. Smullyan. *First-Order Logic.* Springer Verlag, New York, 1968.

[Stickel, 1985] M. E. Stickel. Automated deduction by theory resolution. In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1181–1186, Palo Alto, CA, 1985. Morgan Kaufmann.

[van Arragon, 1991] P. van Arragon. Modeling default reasoning using defaults. *User Modeling and User-Adapted Interaction*, 1:259–288, 1991.

[Weyhrauch, 1980] R. W. Weyhrauch. Prolegomena to a theory of formal reasoning. *Artificial Intelligence*, 13:133–170, 1980.

[Wiebe and Rapaport, 1986] J. M. Wiebe and W. J. Rapaport. Representing *de re* and *de dicto* belief reports in discourse and narrative. *Proceedings of the IEEE*, 74(10):1405–1413, 1986.

[Wilks and Ballim, 1987] Y. Wilks and A. Ballim. Multiple agents and the heuristic ascription of belief. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 118–124, Palo Alto, CA, 1987. Morgan Kaufmann.

[Wilks and Bien, 1983] Y. Wilks and J. Bien. Beliefs, points of view and multiple environments. *Cognitive Science*, 8:120–146, 1983.