Logical Reasoning with Temporal Constraints

Mikael Asker

August 20, 2003

Examensarbete för 20p, Institutionen för Datavetenskap,
Naturvetenskapliga fakulteten, Lunds Universitet

Thesis for a diploma in computer science, 20 credit points,
Department of Computer Science, Faculty of Science, Lund University

Logiska resonemang under tidsbegränsning

Sammanfattning

Logik kan användas för resonemang och kunskapsrepresentation i kunskapsbaserade AI-system. Logiska resonemang har fördelar jämfört med andra former av resonemang genom att det vilar på en stark teoretisk grund. Men logiska resonemang har också en nackdel : de är beräkningskrävande. Dessutom kan det finnas realtidskrav när logiska resonemang används i autonoma agenter. Att utföra beräkningskrävande uppgifter med realtidskrav kan vara en utmaning.

Detta arbete undersöker vad man kan göra åt problemet. Först presenteras några fundamentala teoretiska gränser för tidskraven vid logiska resonemang. Sedan diskuteras en del aktuella metoder för effektiva logiska resonemang. Slutligen föreslås ett nytt angreppssätt: att tillämpa LDS, labelled deductive systems, metodik på aktiv logik.

Logical Reasoning with Temporal Constraints

Abstract

Logic is useful for reasoning and knowledge representation in AI systems. Logical reasoning has an advantage over other forms of reasoning in that it has strong theoretical foundations. But logical reasoning also has a disadvantage: it is computationally difficult. In addition, when logical reasoning is used in physical autonomous agents there might exist real-time requirements. Performing computationally difficult tasks under real-time constraints can be challenging.

This work investigates what can be done to overcome the problem. First some fundamental theoretical limits of time requirements for logical reasoning are presented. Then some current methods for efficient logical reasoning are discussed. Finally a new approach is suggested: applying LDS, labelled deductive systems, methodology to active logic.

Contents

Preface

I want to thank my supervisor Jacek Malec who introduced me to the problems of logical reasoning under time constraints and has answered many questions during the work. The idea of using LDS as a tool to implement time-limited reasoning was originally suggested by Michael Fischer. I also want to thank Sven Asker and Leonard Ilanius for proof-reading my manuscript.

Chapter 1

Introduction

1.1 Problem

Logic is useful for reasoning and knowledge representation in knowledge-based AI systems. Logical reasoning has an advantage over other forms of reasoning in that it has strong theoretical foundations. But logical reasoning also has a disadvantage: it is computationally difficult. In addition, when logical reasoning is used in physical autonomous agents there might exist real-time requirements. Performing computationally difficult tasks under real-time constraints can be challenging.

This work investigates what can be done to overcome the problem. It was done as a Master thesis project at the Computer Science Department at Lund University. My supervisor was Jacek Malec, a member of the AI group at the department.

1.2 Background

Logic was originally invented for purposes quite different from knowledge representation and reasoning in AI systems. The history of logic is long and complex. Logic is a very old subject - Aristotle's syllogisms from 300 BC is a well-known early contribution.

In 1879 Frege began the reshaping of logic from a tool for assessing the validity of arguments in general to a tool for assessing mathematical truth. This was the beginning of a movement, logicism, with the purpose of founding mathematics in logic.

The standard or classical logics of today: propositional logic and predicate or first-order logic, abbreviated FO, came into existence during the first half of the 20th century. For an introduction to standard logic, see [Lem65]. After that, a large number of non-standard logics: higher order, modal, temporal, substructural, relevance, linear, many-valued, ordered, defeasible, non-monotonic, intuitionistic, active, default, epistemic, ... have been developed.

An important step in making logical reasoning possible in computers was the invention of *binary resolution* [Rob65]. Standard first-order logic could now be expressed with a single inference rule operating on formulae in clausal form.

Since then, several new computer-friendly inference rules have been invented - a few of them are mentioned in Section 3.4.

Traditionally a logic was perceived as a consequence relation on a *set* of formulae. Problems arising in some application areas have emphasized the need for consequence relations between *structures* of formulae, such as multisets, sequences or even richer structures. This finer-tuned approach to the notion of a logical system introduced new problems which called for an improved general framework in which many of the new logics arising from computer science applications could be presented and investigated. LDS, *labelled deductive systems*, was presented in [Gab96] as such a unifying framework. It is described briefly in Section 4.5.1.

When used to model the behavior of logical agents, most logics today have a number of shortcomings. [GW01] describes something named "the new logic" which can be used for doing such modelling. They describe it as application of the technical sophistication of mathematical logic to the project of informal logic to get mathematically describable models of what human agents actually do in real-life situations. Formal and informal logic have been a bit separated ever since the formalization for use in mathematics was began in the 19th century - the new logic is an attempt to reunite them.

Compared to logic, AI is a relatively young subject. The work began in the 1940s but the subject got its name at the Dartmouth workshop 1956. See [RN95] for a good introduction to AI.

Logic can be used for knowledge representation and reasoning in knowledge-based AI systems. Some advantages of logic over other forms of knowledge representation are described in [Nil91]. Nilsson advocates "the logical approach to AI". Logicism contra other major approaches to AI is a big question which I will not go into deeper here. But if one has decided to use knowledge-based AI, one can compare logic to other forms of knowledge representation and reasoning.

One argument for using logic is that many smart minds have worked on it for a long time. Logic gives us certain concepts such as entailment, soundness and completeness which turn out to be useful in an AI context. The other alternatives for knowledge representation have similar concepts but the derivation of these concepts are often based on some kind of logic at the bottom level.

One subfield of AI where logic has been used with much success is *automated reasoning*. Applications of automated reasoning include proving mathematical theorems and correctness of programs and digital circuits. Sometimes there is an interesting form of cooperation between the automated reasoning system and the human user. The human user provides guidance and directions based on "intuition" while the automated reasoning system provides ability for massive computation and gurantees the correctness of the results.

One problem with classical logic and many newer forms of logic is that the reasoning is static. Every possible conclusion is supposed to be drawn, as if infinite time was available for the reasoning process. But when implemented in a theorem prover, the inference rules are applied one step at a time and only a finite number of steps is done. *Active logic* is a form of logic that not only allows one to reason *about* time like temporal logic, but where the reasoning process itself is *situated* in time. This makes it possible to reason about the reasoning process itself and it also solves the omniscience problem. More about active logic can be found in Chapter 4.

1.3 Structure of this report

This report is structured as follows:

- Chapter 1: contains this introduction.
- Chapter 2: some fundamental theoretical limits of time requirements for logical reasoning are presented.
- Chapter 3: some current methods for efficient reasoning are discussed.
- Chapter 4: active logic is presented and a new approach is suggested: applying LDS methodology to active logic.
- Chapter 5: contains conclusions.
- Appendix A: two examples of proofs in $\mathbb{L}_{mm}$, one of the LDSes from Chapter 4.

Chapter 2

Fundamental theoretical limits

2.1 Expressiveness versus complexity

Problems formulated in logic are well known to be hard to solve in the worst case. Here are a few examples of computationally difficult problems in logics:

- Determining satisfiability in propositional logic is NP-complete [Pap94].
- Consistency checking in propositional modal logic S4 is PSPACE-complete [SC95].
- As an extreme example, consistency checking in the system LR, relevance logic without the distributivity axiom, is ESPACE-hard - "a computational horror" [GW01] !
- Determining theoremhood and validity in first-order logic is only semidecidable.

Generally speaking, many interesting problems are NP-complete or worse.

The complexity of logical reasoning has been observed in other subject areas than AI. An increase in complexity characterizes the formalization of databases as logical objects versus their characterization as physical objects [SC95].

The use of non-standard forms of logic as tools for knowledge representation in AI makes the computational drawback even larger. Logical formalisms that are used in AI typically have higher complexity than classical logic.

Generally, the more expressive language, the stronger is the logic and the more complex are the corresponding computational problems. There is a trade-off between expressive power and computational tractability in KR formalisms [PS86].

However, tractable logics exist. For example, satisfiability testing of Horn clauses can be done in polynomial time [Pap94]. Fundamental studies on the complexity of fragments of classical propositional and first-order logic have been done in the last decades. More recently, computational studies about several logical formalisms relevant to knowledge representation appeared [SC95]. Studies analyzing the so-called *tractability threshold* between polynomially tractable and intractable laguages are of particular interest here.

2.2 Descriptive complexity

One of the main areas in mathematical logic is model theory. A subfield of that area is *finite model theory*, the study of finite models. An overview of finite model theory is given in [EF99].

The model structures of a theory can be recognized by a Turing machine, and that recognition problem has a certain time complexity. The set of recognition problems of all the theories in a logic has relations to existing complexity classes.

It may be a surprise that some complexity classes are (in some way, as described below) *identical* to the set of recognition problems of some logic. This relation between complexity classes and different kinds of logics is called *capturing*. The field of *descriptive complexity* is the study of such relations. It forms a connection between logic and complexity theory.

The first result of this kind is from [Fag74]:

$$\Sigma_{\mathbf{1}}^{\mathbf{1}} \equiv \mathbf{NP} \tag{2.1}$$

NP, the problems acceptable in nondeterministic polynomial time correspond to $\Sigma_{\mathbf{1}}^{\mathbf{1}}$, the model classes of existential second order logic, whose sentences start with an existential second order prefix followed by a first order kernel.

Capturing results provide time limits for model checking in a logic. The relation between these model checking limits and the time limits for syntactical operations in the logic is non-obvious.

A question that is interesting for us who work with time contraints is: what logics are tractable - decidable in deterministic polynomial time? That question is discussed in [Ebb99]. Ebbinghaus defines three kinds of capturing: weak, strong and effectively strong and proves that there exist logics which weakly and strongly capture **P**. The central question in [Ebb99] is about the effectively strong capturing:

$$\text{Is there a logic } \mathcal{L} \text{ with } \mathcal{L} \equiv_{es} \mathbf{P} \text{ ?} \tag{2.2}$$

That question is important because of its relations to fundamental questions in complexity theory. It will be difficult to get a negative answer, because if the central question 2.2 has a negative answer then $\mathbf{P} \neq \mathbf{NP}$! It will also be difficult to get a positive answer, because if the central question 2.2 has a positive answer, then the graph isomorphism problem is in **P**.

Chapter 3

Some practical methods for efficient reasoning

3.1 Introduction

When constructing an AI system, one or more reasoning tasks might need to be performed in that system. When first formulated, the reasoning problems are typically informal and still only vaguely specified. When a logic is chosen to express such an informal reasoning problem in, one gets a corresponding formal computational problem, as shown in Figure 3.1.

A good choice of logic makes the formal computational problem easier to solve. Such a choice can be achieved by using *language restriction*, which is described in Section 3.2, and by *limiting the inference* using *resource bounded reasoning* and *nontraditional semantics*, which are described in Section 3.3.

Once the choice of logic is fixed, so is the formal computational problem which is bound by the worst-case results stated in Chapter 2. What can be done then is to improve the typical or average case by choosing good *inference rules* and *strategies*. Some ways to do this are described in Section 3.4.

Finally, one more option is to solve the informal reasoning problem by using two different logics "in parallel". Two methods of this kind, *theory approximation* and *knowledge compilation*, are described in Sections 3.5 and 3.6, respec-

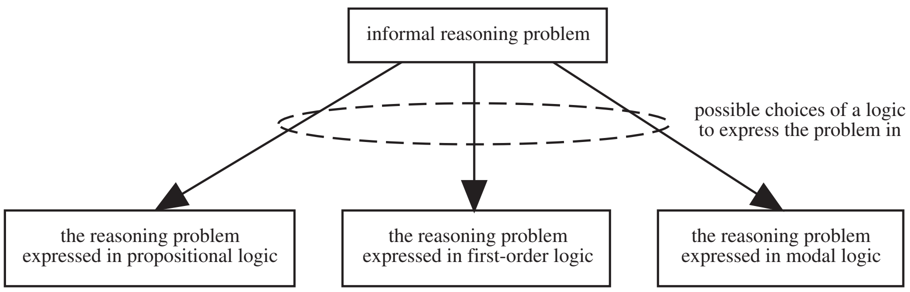


Figure 3.1: Mapping from an informal reasoning problem to a formal computational problem.

tively. Theory approximation provides a way of attacking hard problems that gives indication of their hardness before the correct answer is found. Knowledge compilation can reduce the total amount of work when querying a knowledge base where updates are rare. Another characteristic of knowledge compilation is that most of the computation can be done in advance. This might reduce response time for critical queries.

3.2 Language restriction

One way to reduce the computational load, described in [PS85] and [SC95], is to restrict the language. Choose the logic as weak as possible and the reasoning becomes faster because of the expressiveness-complexity relation described in Section 2.1. The limit of how weak logic that can be used is set by the need to be able to express the informal reasoning problem in the logic.

Language restriction is also discussed in [CD97] in combination with knowledge compilation. This combination is further described in Section 3.6.

3.3 Limited inference

3.3.1 Resource-bounded reasoning

Another way to reduce the computational requirements, described in [SK96], is to limit the inference by bounding the number of steps performed by the inference procedure. Unfortunately, when limiting inference in standard logic it becomes difficult to characterize exactly what can and can not be inferred. The approach lacks a "real" semantics - one that does not simply mimic the proof theory.

A more succesful example of resource-bounded reasoning is the probabilistic inference in [HCH89] where confidence in the result increases with the amount of computation.

3.3.2 Nontraditional semantics

The approach of *nontraditional semantics* limits the inference by weakening the semantics of the language. Like when using language restriction, a logic is chosen to express the informal reasoning problem which gives an easy formal computational problem. But among the logics available to express the reasoning problem in, there might be alternatives which share a common syntax but have different semantics. While keeping the syntax of the logical language fixed, one can sometimes choose between several different semantics of different strength.

One example of using nontraditional semantics is given in [Lev84]. There are deficiencies in current semantic treatments of knowledge and belief. The major existing formal model of belief (due to Hintikka) requires the beliefs of an agent to be closed under logical consequence, and thus can place unrealistic computational demands on its reasoning abilities. This is an example of the problem of logical omniscience, which is described in Section 4.1.

[Lev84] performs a new analysis resulting in a logic that avoids these shortcomings and is also more viable computationally. It uses a weaker sense of belief that is much more attractive computationally and forms a more plausible foundation for the service to be provided by a knowledge representation utility.

All formalizations of belief based on a possible-world semantics suffer from the fact that at any given point, the set of sentences considered to be believed is closed under logical consequence. But there is a much more reasonable way of interpreting the possible-world characterization of belief. Instead of taking logical omniscience as an idealization or heuristic in the modelling of the beliefs of an agent, we can understand it to be dealing realistically with a different though related concept, namely, what is *implicit* in what an agent believes. The proper understanding of a possible-world semantics is that it deals not with what is believed, but what is true given what is believed. We distinguish between *implicit belief* and *explicit belief.*

A better way to formalize the notion of explicit belief is to replace the possible worlds with a different kind of semantic entity that does not necessarily deal with the truth of all sentences. One such entity is *situations* - a generalization of possible worlds where not every sentence in a language is required to have a truth value. A situation may support the truth of some sentences and the falsity of others, but may fail to deal with some other sentences at all. Sentences not relevant to what an agent actually believes need not get a truth value at all. We can think of possible worlds as those limiting cases of situations where every sentence does have a truth value. A better formalization of explicit belief is achieved by identifying it with a set of situations rather than possible worlds.

A second example of using nontraditional semantics is given in [Fri85], where model theory is used to specify the behavior of AI programs. But Frisch's motives for weakening the semantics are different from ours.

In [Fri85] AI programs are viewed as inference engines and their input/output behavior is specified model-theoretically. Intuitions can be specified rigorously and properties of the programs can be proved. The reason for introducing nontraditional semantics is that planning systems like STRIPS result in an incomplete inference engine. By weakening the semantics the inference engine can be made sound and complete.

A third attempt to use nontraditional semantics can be found in [PS85], where nontraditional semantics is used to devise a decidable logics for knowledge representation (KR). Propositional logic is not expressive enough for a KR system. It is generally accepted in KR that the expressive power of at least FOL is needed in a representation language. FOL itself has a semantics that corresponds well with our intuitive ideas about the world. But FOL has a severe problem - determining wether one sentence follows from another is in general undecidable. This makes a KR system built on FOL unsuitable for AI systems that depend on receiving answers.

The syntax of the new logic in [PS85] is the same as that of FOL. However, it is semantically weaker than FOL and because of that all reasoning in it is sound with respect to FOL.

The new logic is contructed by modifying an existing logic, *relevance logic*, which was originally developed as an attempt to solve the paradoxes of material and strict implication. Tautological entailment is an analogue of implication in relevance logic. Relevance logic exists in both propositional and first-order variants and propositional tautological entailment is decidable.

Looking at the first-order variant of relevance logic, first-order situations is the analogue of first-order models. An atomic formula can be assigned true, false, neither or both. But unfortunately, FO entailment is undecidable. This is because quantification is too powerful - it is equivalent to infinite conjunctions or disjunctions.

We would like to have a variant of first-order relevance logic that has a decidable algorithm for determining tautological entailment. In [PS85] this is achieved by giving the quantifiers an intuitionistic reading. Formalization of this change requires a significant modification of FO relevance logic semantics. Instead of talking about situations, sets of situations are needed.

The new logic in [PS85] is considerably weaker than FOL. Its semantics is close to that of standard FO relevance logic. Entailment in the new logic is a decidable subset of implication. But modus ponens does not work which means that chaining does not work.

A fourth example of using nontraditional semantics is given in [PS86], where it is used to make a better frame-based description language. The most important operation when using frame-based description languages is determining if one concept subsumes another. Unfortunately, this is computationally intractable for languages of resonable expressive power. This is the usual trade-off between expressive power and computational tractability. In [PS86] the problem is solved by using a weaker four-valued semantics which gives a smaller set of subsumption relationships.

In recent years there has not been much activity in the area of nontraditional semantics. One reason for this might be problems with understanding and reasoning about what the system really does - a weak semantics can be, and usually is, nonintuitive.

3.4 Inference rules and strategies

Once a logic has been choosen, the computational problem is fixed and bound by the worst-case results presented in Chapter 2. One way to improve the typical or average case is to adjust the inference rules and strategies.

As was mentioned in Section 1.2, Robinson's binary resolution [Rob65] was an important step for achieving logical reasoning in computers. Since then, several new inference rules have been invented. One improved version of binary resolution is *hyperresolution*, which can work on more than two clauses at once. Another improved inference rule is *paramodulation*, which works with equalities.

The improved inference rules complete a proof in fewer, larger steps. But if they are complete they also have a higher branching factor so improved strategies are necessary.

Two strategies which are often used are *unit preference* and *set of support*. They are described in [RN95].

More information about inference rules and strategies can be found in [Wos88].

3.5 Theory approximation

Ideally we would like to get around the expressiveness-complexity trade-off described in Section 2.1 so that we could use an expressive language without a large computational cost. One step in that direction is to try to *approximate* reasoning in a strong language by reasoning in a weaker language. It turns out that such approximation is possible.

Informally, an *approximate solution* to a decision problem is a "maybe" answer, equipped with reasons to believe that the "maybe" is actually a "yes" or to believe that it is actually a "no".

There are two kinds of approximate reasoning: *sound* and *complete*. The sound form has two possible answers: "yes" and "maybe no". If the approximative solution is "yes", then we know for sure that the real answer is "yes". The complete form has the two possible answers "no" and "maybe yes". If the real solution is "yes", then the approximative solution will be "yes". So we have

$$\text{sound ``yes''} \Rightarrow \text{real ``yes''} \tag{3.1}$$

and

$$\text{real ``yes''} \Rightarrow \text{complete ``yes''} \tag{3.2}$$

which means

$$\begin{aligned} \{\text{sound ``yes'' problems}\} &\subseteq \{\text{real ``yes'' problems}\} \\ &\subseteq \{\text{complete ``yes'' problems}\} \end{aligned} \tag{3.3}$$

How do we measure the accuracy of an approximate answer? And how do we know if one approximate answer is any better than another one? In logic we have no explicit metric that gives an immediate answer to the above questions. In this respect, approximation of reasoning problems is more difficult to study than approximation of optimization problems.

An example of theory approximation can be found in [SK96]. Statements in propositional logic, in which the satisfiability problem is intractable, are approximated by *lower and upper Horn bounds.* The Horn bounds are Horn theories, which means that satisfiability testing is tractable.

In the following definition, $\mathcal{M}(\Sigma)$ denotes the set of satisfying truth assignments (models) of the theory Σ:

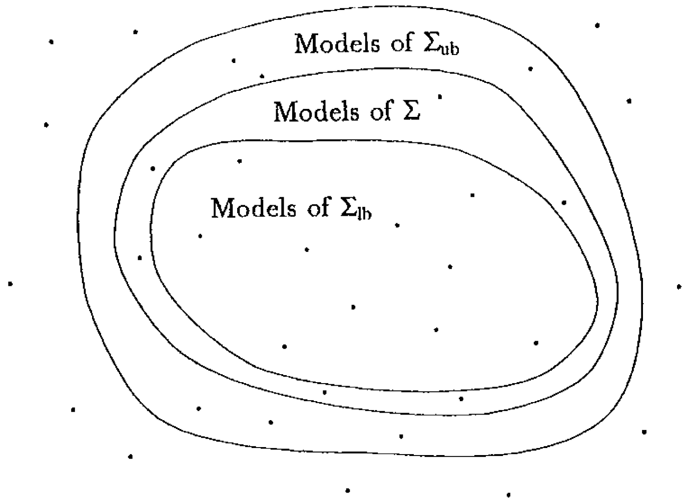


Figure 3.2: Two sets of models of the Horn bounds Σ_{lb} and Σ_{ub} that bound the set of models of the original theory Σ from below and above. Models are represented by dots. Borrowed from [SK96].

Definition 3.1 (Lower and upper Horn bound)
Let Σ be a set of (propositional) clauses. The sets Σ_{lb} and Σ_{ub} of Horn clauses are a *lower* and *upper Horn bound* of Σ, respectively, iff

$$\mathcal{M}(\Sigma_{lb}) \subseteq \mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{ub})$$

or equivalently

$$\Sigma_{lb} \models \Sigma \models \Sigma_{ub}$$

The bounds are defined in terms of models: a lower bound has fewer models than the original theory and an upper bound has more models. Figure 3.2 shows the relations between the set of models of the approximated theory and the sets of models of some lower and upper bounds.

Because the lower bound has fewer models it is logically stronger than (implies) the original theory. The lower bound gives a "yes"/"maybe no" answer and is a complete approximation. Similarly, because the upper bound has more models, it is logically weaker than (implied by) the original theory. The upper bound gives a "no"/"maybe yes" answer and is a sound approximation.

Instead of simply using any pair of bounds to characterize the initial theory, we wish to use the best possible ones:

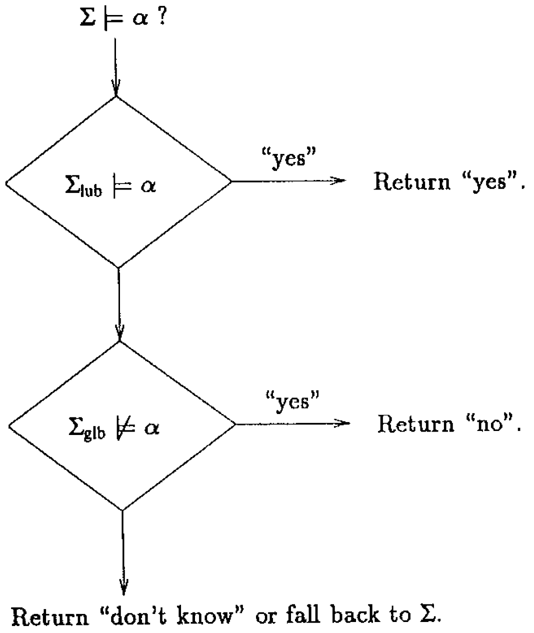


Figure 3.3: Fast querying using the theory approximation by Horn bounds described in [SK96]. The original theory is Σ; Σ_{glb} and Σ_{lub} are its approximations and α is the query. Borrowed from [SK96].

Definition 3.2 (greatest lower Horn bound (GLB))
Let Σ be a set of clauses. The set Σ_{glb} of Horn clauses is a *greatest lower Horn bound* of Σ iff $\mathcal{M}(\Sigma_{glb}) \subseteq \mathcal{M}(\Sigma)$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma_{glb}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$.

Definition 3.3 (least upper Horn bound (LUB))
Let Σ be a set of clauses. The set Σ_{lub} of Horn clauses is a *least upper Horn bound* of Σ iff $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{lub})$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{lub})$.

Figure 3.3 shows how the bounds can be used for fast inference. A knowledge base (KB) contains the set of clauses Σ. We want to determine whether the formula α is implied by the KB. α should be in CNF, because one can determine in linear time if a propositional CNF formula follows from a Horn theory. α itself does not have to be Horn.

The queries that can be answered by the bounds in [SK96] are not always easy to answer using the original theory. Inconsistent theories are obvious examples. Compilation yields inconsistent upper bounds and a query against such bounds quickly returns yes. But evaluating a query against the original theory would in general involve proving that the theory is inconsistent which is coNP-complete.

Another example of queries that can be answered by the bounds but not are easy to answer using the original theory is a consistent theory that is equivalent to a Horn theory but is not in Horn form. *All* queries can be answered efficiently against the bounds. But it is proved in [SK96] that it is *not* the case that a theorem prover can answer queries efficiently against the original theory.

Theory approximation provides a way of attacking hard problems that gives indication of their hardness *before* the correct answer is found.

[SC95] presents the following desiderata list for a theory of approximate reasoning:

semantically well-founded

computationally attractive Approximate answers should be easier to compute than answers to the original problem.

improvable Approximate answers can be improved. The algorithm produces a sequence of approximations which eventually converge to the right answer - provided we have enough time and motivation to compute enough of the steps in the sequence.

dual Both sound and complete approximations should be described.

flexible General enough to be applied to a wide range of reasoning problems.

A reasoning task can be modelled as the problem of deciding wether a string belongs to a set $\mathcal{D}$ (e.g. the set of satisfiable propositional formulae) or not.

The approximation method in [SC95] defines two sequences of sets, $\langle \mathcal{D}_0, \mathcal{D}_1, ..., \mathcal{D}_m \rangle$ and $\langle \mathcal{D}^0, \mathcal{D}^1, ..., \mathcal{D}^n \rangle$, such that

$$\mathcal{D}_0 \subseteq \mathcal{D}_1 \subseteq ... \subseteq \mathcal{D}_m = \mathcal{D} = \mathcal{D}^n \subseteq \mathcal{D}^{n-1} \subseteq ... \subseteq \mathcal{D}^0 \tag{3.4}$$

Improvable approximations can be used to get an anytime algorithm. The motivation to continue the reasoning can be controlled by deliberation scheduling which is described in Chapter 3.7.

The method described in [SC95] is flexible enough to be applied to a wide range of reasoning problems. Approximation techniques are defined for:

- propositional logic;
- fragments of first order logic - concept description languages;
- propositional default logic and circumscription;
- modal logic.

The techniques fulfill the requirements listed above to a varying degree.

Theory approximation is also discussed in [CD97] in combination with knowledge compilation. This combination is further described in the section below.

3.6 Knowledge compilation

Another possibility for performance improvement is based on properties of the typical sequence of operations in some applications. In many knowledge representation scenarios the knowledge base is typically not modified very often but queries are frequent. That property of the sequence of operations can be used by dividing the work in two phases: *preprocessing* and *query answering*. If the knowledge base is kept fixed, the preprocessing phase can be performed offline and only the query answering phase needs to be performed online. Because of this, the phases are sometimes named *offline reasoning* and *online reasoning*.

[CD97] informally defines *knowledge compilation* as "methods of processing a knowledge base off-line in such a way that the output of such a processing can be used to speed up on-line answering for a class of queries. The preprocessing should take a finite amount of time."

The computational cost of the preprocessing phase that can be afforded depends on the frequency of updates. More queries per update means more queries to amortize the compilation cost over.

Most commonly knowledge compilation is used to compile the knowledge base from an intractable form to a tractable one. But, for example, it can make sense to preprocess a PSPACE-complete problem hoping that the residual online problem is "just" NP-complete.

To define when a problem is compilable, [CD97] begins with defining a *structured problem* as consisting of three parts:

- a question P which is the problem itself;
- a fixed part F of the input which can be preprocessed;
- a variable part V of the input which can not be preprocessed.

Compilability can now be defined as follows:

Definition 3.4 (Compilability)
A structured problem (P, F, V) is *compilable* if there exist two polynomials p_1 and p_2 and an algorithm ASK such that for each instance f of the fixed part F there is a compiled data structure D_f such that
1. $|D_f| \leq p_1(|f|)$.
2. for each instance v of the variable part V the call $ASK(D_f, v)$ returns yes iff (f, v) is a "yes" instance of the question P.
3. $ASK(D_f, v)$ requires time $p_2(|v| + |D_f|)$.

To give a formal proof that a problem is *not* compilable is quite difficult. Typically, if the problem was compilable, that would imply things like $\Sigma_\mathbf{2}^\mathbf{p} = \Pi_\mathbf{2}^\mathbf{p}$ (the polynomial hierarchy collapses) or $\mathbf{P} = \mathbf{NP}$. A parallel is that proving polynomial-time intractability is usually done by proving **NP**-hardness, which means that a problem does not have polynomial-time algorithms provided $\mathbf{P} \neq \mathbf{NP}$.

vars	clauses	bounds and *tableau*		*tableau* only	
		binary	ternary	binary	ternary
75	322	51	48	258	248
100	430	54	45	368	341
150	645	61	59	1286	1084
200	860	55	51	12962	8632

Table 3.1: Results from [SK96]: time in seconds to answer 1000 random queries using the bounds together with the program *tableau* (a version of the Davis-Putnam algorithm) versus using *tableau* alone. All experiments were done on a 100 MHz SGI Challenge workstation. The Davis-Putnam procedure was at the time when the experiments were performed the fastest known complete procedure for propositional satisfiability testing and theorem proving on the class of formulae considered. *tableau* was one of the fastest implementations of the algorithm.

Knowledge compilation can be combined with language restriction as described in [CD97]. It can also be combined with theory approximation. Compilation methods can be grouped in two classes: *approximative compilation* which contain an approximation part and *exact compilation*, which does not include an approximation part.

An example of approximative compilation where propositional formulae are compiled to Horn clauses is given in [SK96]. The Horn bounds used there have already been described in Section 3.5.

The compilation method in [SK96] was tested with hard randomly generated propositional theories. Most randomly generated theories are easy to reason with. Such theories tend to be either very overconstrained or very underconstrained. But computationally challenging theories can be generated randomly by generating formulae with a particular ratio of clauses to variables. For random 3CNF formulae, the ratio is about 4.3.

The results from testing with randomly generated theories are shown in Table 3.1. The performance gain is dramatic and increases with the number of variables and clauses in the theory.

3.7 Deliberation scheduling

[BD94] describes *deliberation scheduling*, a method for controlling decision making procedures such as planning and problem solving, which might include logical reasoning. The decision making procedures are performed in a time-constrained environment: an agent with limited computational capabilities. The agent is embedded in a complex environment with other processes and agents which are not under its control. The amount of time spent on decision making affects the quality of the system's responses. Expectations about the performance of the decision making procedures and the preferred outcomes are provided as input to the scheduling. The scheduling then explicitly allocates computational resources to the decision making procedures based on the expected effect on system performance.

[HCH89] describes a form of deliberation scheduling based on probabilistic inference and demonstrates its use in medical applications.

Chapter 4

Active logic

4.1 Introduction

Several problems arise when using logic in autonomous agents. One of them is *logical omniscience*. A "perfect" logical reasoner can deduce any valid conclusion instantaneously. When it has a belief it also automatically believes all logical consequences of that belief. This is *logical omniscience* and is not realistic.

Another problem when using logic in autonomus agents is handling contradictions. When a classical reasoning system encounters a contradiction it can derive all well-formed formulae as theorems from that contradiction. This is known as the *swamping problem*.

An agent operating in the real world never has complete knowledge. When reasoning with incomplete knowledge, assumptions are often made which have to be retracted later because they are contradicted by new knowledge. One way to make assumptions is *default reasoning*, a simple form of nonmonotonic reasoning which can be handled well by active logic.

An overview of active logic is given in [EDKM$^+$99].

Active logic has some promising properties concerning biological and psychological realism:

- The size of STM as described in Section 4.2;
- Focus of attention;
- Ability to solve the omniscience problem;
- Ability to solve the swamping problem;
- Efficient nonmonotonic reasoning;
- An endless stream of conclusions rather than a final result.

The price we have to pay for this is unusual semantics. Conventional non-active logics works with the final tray concept - every possible conclusion is supposed to be drawn, as if there were infinite time available for the reasoning process. In active logic the set of derived formulae grows with time. Another

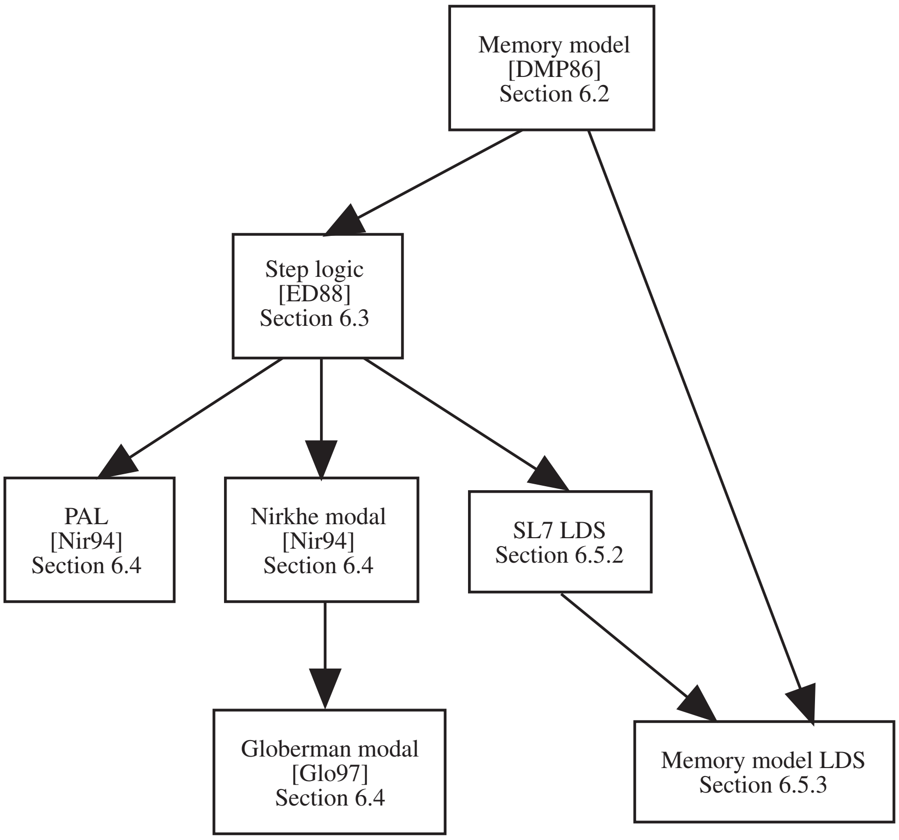


Figure 4.1: The relationships between the kinds of active logic described in this chapter.

difference is that active logics produce an endless stream of conclusions rather than a final result - the reasoning never stops but is an ongoing process.

Figure 4.1 shows the relationships between the different kinds of active logic which are described in this chapter.

In this chapter we will first take a look at the memory model behind active logic. We will then look at *step logic*, the original form of active logic, and some other forms of active logic which were developed later. Finally we will try to describe active logic using the powerful LDS (labelled deductive systems) formalism.

4.2 A memory model inspired from cognitive psychology

Active logic started as an attempt to formalize a memory model inspired by cognitive psychology research which was studied at the University of Maryland during the 1980s [DMP86]. Two different direct formalizations of that model are described here, *step logic* in Section 4.3 and my own LDS solution in Section 4.5.3.

This section describes the model itself. It consists of five parts:

- LTM, the *long term memory*, which contains rules. Semantic retrieval is associative based on trigger formulae.
- STM, the *short term memory*, which acts as the current focus of attention. All new inferences must include a formula from the STM.
- QTM, the *quick term memory*, which is a technical device for buffering the next cycle's STM content.
- RTM, the *relevant term memory*, which is the repository for default reasoning and relevance. It contains formulae which have been pushed out of the STM but still may be important for default resolution.
- ITM, the *intermediate term memory*, which contains all facts which have been pushed out of the STM. The contents of the ITM provides the history of the agents reasoning process. ITM provides support for goal-directed behavior.

Three of the parts, LTM, STM and ITM, come from cognitive pychology. The other two, QTM and RTM, have been invented by Drapkin, Miller and Perlis. Figure 4.2 shows how the parts are connected to each other.

The memory model is demonstrated to work on some examples of simple default reasoning.

[DMP86] contains an interesting observation about the size of the short term memory. Experimentation has shown that a size of roughly eight is the smallest that leads to effective task-oriented behavior over several domains, and that larger sizes offer no advantage. This is in surprising accord with psychological data on human short-term memory which has been measured to hold seven plus or minus two "chunks" of data at any one time. It might be pure coincidence, but it may also indicate some important similarity between the memory model and the human short-term memory mechanism.

4.3 Step logic

The first form of active logic was *step logic* which is described in [EDP88], [ED88] and [EDMP91]. It was an attempt to formalize parts of the memory model described in the previous section. Another, more complete approach to formalization of that model will be given in Section 4.5.3.

Step logic is not a single system but a family of different step logics. Each of the step logics in the family contains two distinct types of formalisms: the *metatheory SL^n about* the reasoning agent and the *agent theory SL_n* itself, which is step-like. The two theories together form a *step-logic pair* $< SL_n, SL^n >$.

The subscript n serves to distinguish different versions of the step logics. The versions differ in the mechanisms that the agent has at its disposal: self knowledge, time and retraction. Self knowledge gives the agent the capability to introspect and determine what it does and does not know. The time mechanism

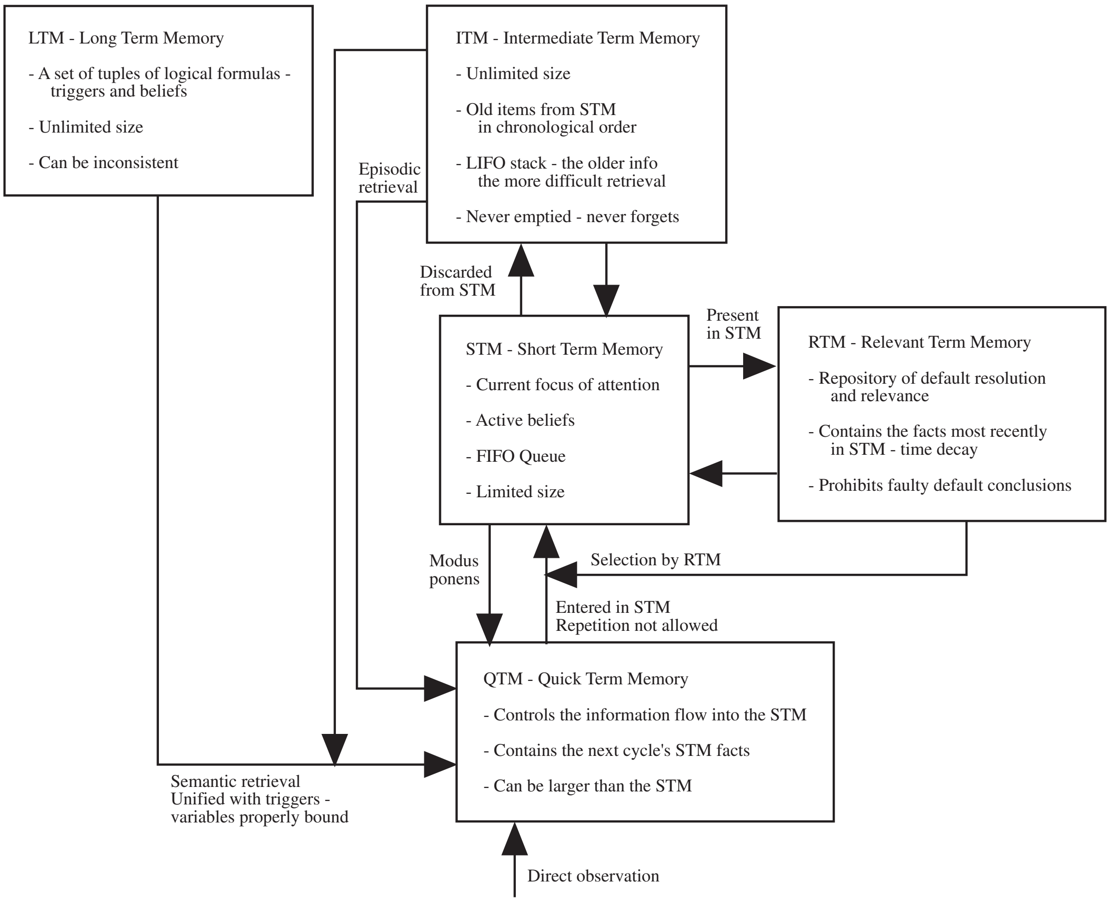


Figure 4.2: The memory model from [DMP86].

allows the on-going process of deduction to be part of the agent's own reasoning. Retractions can be used to implement nonmonotonic reasoning.

Elgot-Drapkin proposed eight step-logic pairs, arranged in increasing sophistication with respect to the three mechanisms above. The agent theories and the mechanisms that they included were as follows (S = self knowledge, T = time, R = retraction):

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

Beliefs are parameterized by the time taken for their inference.

In the following definitions, $\mathbb{N}$ (the set of natural numbers) is used as a model of discrete time. S_{wff} is the set of all well-formed formulae of the language of predicate logic. $\mathcal{P}(S)$ denotes the power set of the set S.

Definition 4.1 (Observation function)
An *observation function* is a function $OBS : \mathbb{N} \rightarrow \mathcal{P}(S_{\mathit{wff}})$, where for each $i \in \mathbb{N}$ the set $OBS(i)$ is finite. If $\alpha \in OBS(i)$, then α is called an i-observation.

Definition 4.2 (History)
A *history* is a finite tuple of belief set/observation set pairs; the sets are finite subsets of S_{wff}:

$$<< \mathit{Thm}_0, OBS(1) >, < \mathit{Thm}_1, OBS(2) >, \ldots ,$$
$$\ldots , < \mathit{Thm}_{i-1}, OBS(i) >>$$

$\mathcal{H}$ denotes the set of all histories.

Intuitively, a history is a conceivable temporal sequence of belief set/observation set pairs.

The *inference function* extends the temporal sequence of belief sets by one more step beyond the history:

Definition 4.3 (Inference function)
An *inference function* is a function $INF : \mathcal{H} \rightarrow \mathcal{P}(S_{\mathit{wff}})$, where for each $h \in \mathcal{H}$, $INF(h)$ is finite.

We can now define the first type of formalism, the agent theory:

Definition 4.4 (SL_n-theory)
An *SL_n-theory* over a language $\mathcal{L}$ is a triple $< \mathcal{L}, OBS, INF >$, where $\mathcal{L}$ is a first order language, OBS is an observation function and INF is an inference function. We use the notation $SL_n(OBS, INF)$ for such a theory (the language $\mathcal{L}$ is implicit in the definitions of OBS and INF). If we wish to consider a fixed INF but a variable OBS, we write $SL_n(\cdot, INF)$.

Members of the set S_{wff} of well-formed formulae over the language $\mathcal{L}$ are called *agent wffs*. SL_n-theories will often be called *agent theories*.

Definition 4.5 (i-theorem, $\vdash_i$)
Let the set of *0-theorems*, denoted Thm_0 be empty. For $i > 0$, let the set of *i-theorems*, denoted Thm_i, be $INF(<< Thm_0, OBS(1) >, < Thm_1, OBS(2) >, ..., < Thm_{i-1}, OBS(i) >>)$. We write $SL_n(OBS, INF) \vdash_i \alpha$ to mean α is an i-theorem of $SL_n(OBS, INF)$.

Intuitively, α is an i-theorem if it can be inferred in i steps from the observations.

It is now time to define the second type of formalism, the meta-theory:

Definition 4.6 (Meta-theory corresponding to SL_n)
Given a theory $SL_n(OBS, INF)$, a corresponding SL^n*-theory*, written $SL^n(OBS, INF)$, is a first-order theory having binary predicate symbol K, numerals and names for the wffs in S_{wff}, such that

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

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

The predicate letter K has two roles: in SL^n and in SL_n. In the agent theory it refers to constant symbols which are *names* of agent wffs.

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

Definition 4.7 (Analytical completeness)
A meta-theory SL^n is *analytically complete*, if for every positive integer i, and every constant α naming an agent wff of the corresponding agent theory, either $SL^n \vdash K(i, \alpha)$ or $SL^n \vdash \neg K(i, \alpha)$.

SL^0 is proved to be analytically complete in [ED88].

Definition 4.8 (Step interpretation)
Let $\mathcal{L}'$ be a language having the symbols of $\mathcal{L}$ and the (possibly additional) predicate symbols K and Now. Thus $\mathcal{L}'$ may be $\mathcal{L}$ itself. A *step-interpretation* for $\mathcal{L}'$ is a sequence $M =< M_0, M_1, ..., M_i, ... >$, such that
1. Each M_i is an ordinary first-order interpretation of $\mathcal{L}'$.
2. $M_i \models Now(i)$.

Definition 4.9 (Step model)
A *step model* for $SL_n(OBS, INF)$ is a step-interpretation M satisfying for all $i \in \mathbb{N}$
1. $M_i \models K(i, \alpha)$ iff $SL_n(OBS, INF) \vdash_i \alpha$.
2. $M_i \models \alpha$ whenever $SL_n(OBS, INF) \vdash_i \alpha$.

Condition 1 ensures that a historical record of the i-theorems exists. Condition 2 ensures that the i-theorems are in fact true.

Definition 4.10 (i-truth in a step model)
A wff α is *i-true in a step model M* (written $M \models_i \alpha$) if $M_i \models \alpha$.

Definition 4.11 (Stepwise consistent theory)
$SL_n(OBS, INF)$ is *stepwise consistent* if for each $i \in \mathbb{N}$, the set of i-theorems is consistent.

Even if $SL_n(OBS,INF)$ is stepwise consistent, it can have conflicting wffs at different steps. For example, $SL_n(OBS, INF) \vdash_{10} Now(10)$ and $SL_n(OBS, INF) \vdash_{11} \neg Now(10)$.

Definition 4.12 (Eventually consistent theory)
$SL_n(OBS, INF)$ is *eventually consistent* if $\exists i \forall j > i$, the set of j-theorems is consistent.

Any stepwise consistent theory is eventually consistent.

Definition 4.13 (Finite observation function)
An observation function OBS is *finite* if $\exists i \forall j > i, OBS(j) = \emptyset$.

Definition 4.14 (Self-stabilizing theory)
$SL_n(\cdot, INF)$ is *self-stabilizing* if for every finite OBS, $SL_n(OBS, INF)$ is eventually consistent.

Intuitively, a self-stabilizing theory $SL_n(\cdot, INF)$ corresponds to a fixed agent that can regain and retain consistency after being given arbitrarily (but finitely) many contradictory initial beliefs.

Theorem 4.1 (Soundness - proved in [ED88])
Every step-logic $SL_n(OBS, INF)$ is sound with respect to step-models. That is, every i-theorem α of $SL_n(OBS, INF)$ is i-true in every step-model M of $SL_n(OBS, INF)$, i.e., if $SL_n(OBS, INF) \vdash_i \alpha$ then $M \models_i \alpha$.

In [ED88], step logic is tested on two well-known problems: the brother problem and the wise-men problem.

A problem with step logics is that the set of beliefs grows rapidly. When the memory model in Section 4.2 was formalized in step logic, some of its properties were lost. In the memory model the STM simulates a focus of attention. The size limit of the STM limits the number of inferences per step and avoids combinatorial explosion which is important in implementations. A vague analogy is the set of support strategy mentioned in Section 3.4. In step logics the limitation is lost so that the number of formulae in each step increases rapidly. This problem has been recognized in [NKP93], [Nir94] and [Glo97]. An attempt to improve this is presented below in Section 4.5.3.

4.4 Other active logics

In [Nir94], active logic is used in an application: planning with tight deadlines. An active logic named PAL - Planning Active Logic - is used for the reasoning. It has some inference rules that are domain-independent but are specific to planning and acting in deadline sutuations. PAL uses a different kind of STM than the one in [DMP86]. The PAL STM is not organized as a FIFO queue. PAL is tested on two well-known problems, the Nell and Dudley problem and the Yale Shooting Problem.

Section 8.5 of [Nir94] contains a proof that an STM of limited size is sufficient for monotonic reasoning. With an STM of limited size there is always an algorithm for fetching elements into the STM such that any theorem in the original step logic with unlimited memory will also appear as a theorem in the limited size STM version.

[Nir94] also points out a gap between reasoning with knowledge in modal logics and the time-situated approach of step logics. An attempt to bridge this gap is done by using a modal semantics based on the semantics from Montague's intensional logic of belief, which avoids the omniscience problem associated with the standard possible world semantics. The omniscience problem is the same as the one solved in [Lev84] in Section 3.3.2, but this solution is completely different.

Yet another active logic is the active modal logic described in [Glo97], which uses focus and belief operators.

4.5 Active logic and LDS

4.5.1 An introduction to LDS

Traditionally a logic was perceived as a consequence relation on a *set* of formulae. Problems arising in some application areas have emphasized the need for consequence relations between *structures* of formulae, such as multisets, sequences or even richer structures. This finer-tuned approach to the notion of a logical system introduces new problems which call for an improved general framework in which many of the new logics arising from computer science applications can be presented and investigated. LDS, *labelled deductive systems*, was presented in [Gab96] as such a unifying framework.

The first step in understanding LDS is to understand the intuitive message, which is very simple. Traditional logics manipulate formulae. An LDS manipulates *declarative units*, pairs *formula* : *label* of formulae and labels. This sounds very simple, but turns out to be a big step which makes a serious difference, like the difference between using one hand only and allowing for the coordinated use of two hands.

The labels should be viewed as more information about the formulae, which is not encoded inside the formulae. For example: reliability in an expert system, where and how a formula was deduced, time stamps.

LDS is a methodology, not a single system. Gabbay recommends usage of LDS only if it is convenient for the application. It should be used to simplify, not to complicate. The user should be guided by the application and its needs.

A *logic* is a pair $(\vdash, S_\vdash)$ where $\vdash$ is a structured, possibly nonmonotonic consequence relation on a language L and $S_\vdash$ is an LDS. $\vdash$ is essentially required to satisfy no more than identity (i.e. $\{A\} \vdash A$) and a version of cut.

A simple form of LDS is the *algebraic LDS*. There are more advanced variants, *metabases*, in which the labels can be databases.

An *LDS proof system* is a triple $(\mathcal{A}, \mathbf{L}, \mathbb{R})$ where $\mathcal{A}$ is an algebra of labels (with some operations), $\mathbf{L}$ is a logical language (connectives and wffs) and $\mathbb{R}$ is a discipline of labelling formulae of the logic (with labels from the algebra A), together with a notion of a *database* and a family of deduction rules and with agreed ways of propagating the labels via the application of the deduction rules. The algebra $\mathcal{A}$ can be described by a labeling language, which is a logical language in itself.

A proper *practical reasoning system* has 'mechanisms' for updates, inputs, abduction, actions, etc., as well as databases (theories, assumptions). Gabbay's personal view is that a proper logic is an LDS system integrated together with a specific choice of such mechanisms. In AI circles this might be called an agent.

The traditional logic community is still very conservative in the sense that it has not even accepted nonmonotonic reasoning systems as logic yet. It believes that all this excitement is transient, temporarily generated by computer science and that it will disappear sooner or later. It believes that we will soon be back to the old research problems, such as how many non-isomorphic models does a theory have in some inaccessible cardinal or what is the ordinal of yet another subsystem of analysis. Gabbay thinks this is fine for mathematical logic but not for the logic of human reasoning. There is no conflict here between the new and the old, just further evolution of the subject.

Algebraic LDS has its own proof theory based on some standard types of inference rules and its own semantics. Metabases have a special kind of semantics, fibred semantics. [Gab96] investigates the relation between traditional logics and LDS and proves that is possible to translate any LDS to FO.

[Gab96] contains many examples of formulating major known logics in LDS. An example of using LDS to analyze substructural logics is given in [DG94] and is also included in Chapter 10 of [Gab96].

LDS has not yet been used in the mainstream of AI research. It is a powerful methodology but perhaps a bit difficult to understand in depth.

We will now try to describe two forms of active logic with LDS. We have not been able to find any earlier attempt to do this in the literature. Hopefully it can help us to make a better formalization of the memory model and get rid of some of the limitations of step logic which were mentioned in Section 4.3.

4.5.2 $SL_7(\cdot, INF_B)$ as an LDS

As a first step of our attempt to faithfully formalize the memory model of [DMP86], we express the most complex agent theory from the step logic approach, SL_7, as an LDS. This illustrates the adopted strategy without a necessity to introduce the details required to fully formalize the memory model. However, the next section is going to present exactly those details.

The *observation function*, $OBS : \mathbb{N} \to \mathcal{P}(S_{wff})$, where S_{wff} is the set of well-formed formulae of SL_7 and $\mathcal{P}(X)$ denotes the power set of a set X, is used to generate the axioms. For every time point $i \in \mathbb{N}$, $OBS(i)$ is finite.

As labels we use the natural numbers that represent the time at which a formula has been asserted (observed or deduced); simply, $S_{labels} \overset{df}{=} \mathbb{N}$. The *declarative units* of the system are pairs *label* : *formula*:

$$S_{du} \overset{df}{=} S_{labels} \times S_{wff} \tag{4.1}$$

Sometimes, to improve readability, we use the notation $\boxed{label : formula}$, instead of just *label* : *formula*, where the box serves purely as a delimiter and has no additional meaning by itself.

The axioms below (actually, axiom schemata) express either the time flow or results of observations. At each time point i the formula $Now(i)$ is true. If OBS is the observation function then at every time point i we assert observations associated with this time point.

(A1)	$i : Now(i)$	for all $i \in \mathbb{Z}_+$	Clock
(A2)	$i : \alpha$	for all $\alpha \in OBS(i),\ i \in \mathbb{Z}_+$	Observations

Except in the clock axioms the *Now* predicate has no special status in the system. Just like in step logic it is part of the time mechanism and it should be used in applications just like in step logic.

The inference rules used in the system, $\mathbb{R}_{SL_7}$, come from INF_B defined in [EDP88], although in somewhat modified form in order to take account of labels. First come two versions of Modus Ponens:

(I1) $$\frac{i : \alpha, i : \alpha \to \beta}{i+1 : \beta}$$ Modus Ponens

(I2) $$\frac{i : P_1 a, \ldots, i : P_n a, i : (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \to Qx]}{i+1 : Qa}$$ Extended Modus Ponens

The next rule, Negative Introspection, allows one to infer lack of knowledge of a particular formula at time i. In order to express that we need to define the set

$S_{th}(i)$ of conclusions that can be drawn at time i. We begin with the set of all axioms, S_{axioms}, obtained from (A1) and (A2). Then we look for all formulae that can be derived from it using the rules of inference. Finally, we choose out those that have the label i in front and call them *i-theorems*:

$$S_{th}(i) \stackrel{df}{=} \left\{\boxed{j : \alpha} \in S_{du} \mid S_{axioms} \vdash \boxed{j : \alpha} \wedge j = i\right\} \tag{4.2}$$

$S_{th}(i)$ can be computed by purely syntactical operations and it can be defined recursively using the inference rules. It is well-defined for every $i \in \mathbb{N}$ because the consequence relation is "directed" by the natural ordering of the set $\mathbb{N}$. Every inference rule necessarily increments the label. Therefore all the elements in $S_{th}(i)$ will be inferred from a finite number of instances of axiom (A1), namely those for which labels vary between 0 and $i-1$, and from the finite amount of observations performed until the time i. As every inference rule increments the label, only a finite number of applications of every rule is possible before the label reaches i.

Given a finite set $S_{th}(i)$ of i-theorems, we can identify all closed subformulae ocurring in them and not occuring as separate theorems. The process of finding all closed subformulae for a given finite set of formulae is computable. We begin with the "is a closed subformula" relation $\mathcal{R}_{csf} \subseteq S_{wff} \times S_{wff}$, namely

$$\alpha \, \mathcal{R}_{csf} \, \beta \Leftrightarrow \alpha \text{ is a closed subformula of } \beta \tag{4.3}$$

We define the function f_{csf}, $f_{csf} : \mathcal{P}(S_{du}) \to \mathcal{P}(S_{wff})$, constructing (unlabeled) closed subformulae out of a given set of (labeled) formulae $S \in \mathcal{P}(S_{du})$:

$$f_{csf}(S) \stackrel{df}{=} \{\alpha \in S_{wff} | (\exists \boxed{i : \beta} \in S)(\alpha \, \mathcal{R}_{csf} \, \beta)\} \tag{4.4}$$

Finally, the (projection) function $f_{formulas}$, $f_{formulas} : \mathcal{P}(S_{du}) \to \mathcal{P}(S_{wff})$ extracts unlabeled formulae out of a set $S \in \mathcal{P}(S_{du})$ of labeled formulae:

$$f_{formulas}(S) \stackrel{df}{=} \{\alpha \in S_{wff} | (\exists i \in S_{labels})(\boxed{i : \alpha} \in S)\} \tag{4.5}$$

We can now formulate the Negative Introspection rule:

$$\text{(I3)} \quad \frac{\alpha \in f_{csf}(S_{th}(i)), \alpha \notin f_{formulas}(S_{th}(i))}{i+1 : \neg K(i, \alpha)} \qquad \begin{array}{c}\text{NEGATIVE}\\ \text{INTROSPECTION}\end{array}$$

The agent is supposed to be aware of all the closed subformulae in $S_{th}(i)$. They provide a relevant and finite subset of S_{wff} for the self-knowledge mechanism to operate on.

Except in rule (I3) the K predicate has no special status in the system. Just like in step logic it is part of the self-knowledge mechanism and it should be used in applications just like in step logic.

The contradiction detection works just like in step logic:

$$\text{(I4)} \quad \frac{i : \alpha, i : \neg\alpha}{i+1 : \mathit{Contra}(i, \alpha, \neg\alpha)} \qquad \begin{array}{c}\text{CONTRADICTION}\\ \text{DETECTION}\end{array}$$

Conventional logical systems breaks down when given a contradiction because of the swamping problem - every formula can be derived from a contradiction. But step logic and this LDS is not bothered by implicit contradictions. Direct contradictions of the form $\alpha, \neg\alpha$ are detected by this rule and the *Contra* formula can initiate some kind of clean up.

The inheritance rule propagates the knowledge in time, from one time step to the next. It handles the detected contradictions and prevents the clock axioms from being inherited:

$$\text{(I5)} \quad \frac{\begin{array}{l} i : \alpha \\ \mathit{Contra}(i-1, \alpha, \beta) \notin f_{\mathit{formulas}}(S_{\mathit{th}}(i)) \\ \mathit{Contra}(i-1, \beta, \alpha) \notin f_{\mathit{formulas}}(S_{\mathit{th}}(i)) \\ \alpha \neq \mathit{Now}(i) \end{array}}{i+1 : \alpha} \quad \textsc{Inheritance}$$

We can now define the LDS which is intended to express the SL_7 agent theory:

> **Definition 4.15 (SL_7 LDS)**
> $\mathbb{L}_{SL_7} \stackrel{df}{=} (\mathbb{N}, \mathbf{L}, \mathbb{R}_{SL_7})$, where $\mathbb{R}_{SL_7}$ is built around (I1)–(I5).

where $\mathbb{N}$ is interpreted as an algebra.

A database $\Delta_{ax}(OBS)$ containing the declarative units in S_{axioms} can be generated from the observation function OBS with another function Δ_{ax}.

It is now time to prove that $\mathbb{L}_{SL_7}$ works as expected:

> **Theorem 4.2 (Correctness of $\mathbb{L}_{SL_7}$)**
> A formula α can be derived in the step logic $SL_7(OBS, INF_B)$ at time point i iff it can be derived as $\boxed{i : \alpha}$ in the labelled deductive system $\mathbb{L}_{SL_7}$ defined above. Another way to put it is that for all observation functions OBS, points in time i, and well-formed formulae α:
>
> $$SL_7(OBS, INF_B) \vdash_i \alpha \Leftrightarrow \Delta_{ax}(OBS) \vdash_{\mathbb{L}_{SL_7}} \boxed{i : \alpha}$$

Proof:
The statement above is the same as

$$\alpha \in \mathit{Thm}_i \Leftrightarrow \boxed{i : \alpha} \in S_{th}(i)$$

which is the same as

$$\mathit{Thm}_i = f_{\mathit{formulas}}(S_{th}(i)).$$

That can be proved by induction over time: Thm_0 and $S_{th}(0)$ are empty by definition so $\mathit{Thm}_i = f_{\mathit{formulas}}(S_{th}(i))$ is true for $i = 0$. Assume that

$Thm_i = f_{formulas}(S_{th}(i))$ is true for i. By definition

$$\begin{aligned} Thm_{i+1} = INF_B(<\ & < Thm_0, OBS(1) >, \\ & < Thm_1, OBS(2) >, \\ & ..., \\ & < Thm_i, OBS(i+1) >>) \end{aligned}$$

INF_B applies its inference rules to Thm_i and "adds" the result of this to the observations in $OBS(i+1)$. In the LDS $\mathbb{L}_{SL_7}$ we get from $S_{th}(i)$ to $S_{th}(i+1)$ by applying the $\mathbb{R}_{SL_7}$ rules to the declarative units in $S_{th}(i)$ and "adding" the result of this to the axioms with label $i+1$. The resulting unlabeled formulae are the same in both systems. This can be found by comparing the inference rules one by one. □

4.5.3 The memory model as an LDS

In our opinion the formalisation of the memory model in step logic (see Section 4.2) is an oversimplification. In particular, in the memory model the short-term memory (STM) simulates the focus of attention. The limited size of the STM limits the number of inferences per step and allows to avoid the combinatorial explosion which is important in implementations. A vague analogy is the set of support strategy mentioned in Section 3.4. In step logics this limitation is omitted so that the number of formulae in each step may increase rapidly. This problem has also been recognized in [NKP93], [Nir94] and [Glo97].

Our next step is to extend the SL_7 LDS to include all aspects of the memory model from [DMP86]. The resulting LDS is presented below. In order to encompass all the complexity of the model, the labels here need to be more complex as well. Namely,

$$S_{labels} \stackrel{df}{=} \{LTM, QTM, STM, ITM\} \times S_{wff} \times \{C, U\} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \quad (4.6)$$

where the interpretation of a tuple in S_{labels} is the following. If

$(location, trigger, certainty, time, position, time\text{-}left\text{-}in\text{-}rtm) \in S_{labels}$

is a label, then *location* encodes the memory bank location of the formula (one of LTM, QTM, STM or ITM), *trigger* is used for encoding the triggering formula for LTM items (in particular, ε is used to denote the empty triggering formula), *certainty* is used in case of defeasible reasoning to encode the status of the formula (certain or uncertain), *time* is the inference time, *position* denotes the formula's position in STM or ITM, and, finally, *time-left-in-rtm* denotes the time the labelled formula should remain in the RTM.

The RTM has no *location* value. A labelled formula is present in RTM and available for resolving contradictions when its *time-left-in-rtm* field is non-zero. $R \in \mathbb{N}$ is a constant used to limit the time a formula remains in RTM after it has left STM. The inference rules enter a formula into RTM by setting *time-left-in-rtm* to R when the formula enters STM. *time-left-in-rtm* remains at R until the formula leaves STM. When the formula has left STM and moved to ITM, *time-left-in-rtm* is decremented by one at each time step until it reaches zero. When *time-left-in-rtm* reaches zero, the formula leaves RTM.

The set of axioms, S_{axioms}, is determined by the following two axiom schemata:

(A1')	$(STM, \varepsilon, C, i, i, 0) : Now(i)$	for all $i \in \mathbb{Z}_+$	CLOCK
(A2')	$(QTM, \varepsilon, C, i, 0, 0) : \alpha$	for all $\alpha \in OBS(i),\ i \in \mathbb{Z}_+$	OBS

and the following finite set of axioms

(A3)	$(LTM, \alpha, c, 0, 0, 0) : \beta$	for all trigger - certainty level - formula tuples (α, c, β) present in LTM at time 0	LTM

The rules of inference describe not only which formulae may be derived from others but also the memory banks for the source and result formulae. The first rule describes retrieval from the LTM into QTM:

(SR) $$\frac{(STM, \varepsilon, c_1, i, p, R) : \alpha, (LTM, \beta, c_2, i, 0, 0) : \gamma, \alpha\ \mathcal{R}_{sr}\ \beta}{(QTM, \varepsilon, c_2, i, 0, 0) : \gamma}$$ SEMANTIC RETRIEVAL

The relation $\mathcal{R}_{sr}$ describes how the trigger formulae control the semantic retrieval. The trigger formulae in the LTM labels and $\mathcal{R}_{sr}$ together limit the flow of information from LTM to QTM. It is important to limit that flow, especially in this implementation where there is little selection when the information passes from QTM to STM, see below. In the examples we have used $\mathcal{R}_{sr} = \mathcal{R}_{csf}$.

The "real" inference using Modus Ponens is performed from STM to QTM:

(MP) $$\frac{(STM, \varepsilon, c_1, i, p_1, R) : \alpha, (STM, \varepsilon, c_2, i, p_2, R) : \alpha \rightarrow \beta}{(QTM, \varepsilon, min(c_1, c_2), i, 0, 0) : \beta}$$ MODUS PONENS

(EMP) $$\frac{\begin{array}{l}(STM, \varepsilon, c_1, i, p_1, R) : P_1 a \\ \dots \\ (STM, \varepsilon, c_n, i, p_n, R) : P_n a \\ (STM, \varepsilon, c_{n+1}, i, p_{n+1}, R) : (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Qx]\end{array}}{(QTM, \varepsilon, min(c_1, \ldots, c_{n+1}), i, 0, 0) : Qa}$$ EXTENDED MODUS PONENS

where function min is defined over the set $\{U, C\}$ of certainty levels, with the natural ordering $U < C$. The idea behind it is that the status of a consequence should not be stronger than any of its premises. It is a mechanism that allows us to introduce a simple form of non-monotonicity.

We need one more rule to handle defeasible inference, namely Negative Introspection:

(NI) $$\frac{\alpha \in f_{csf}(S_{STM}(i)), \alpha \notin f_{formulas}(S_{STM}(i))}{(QTM, \varepsilon, C, i, 0, 0) : \neg K(i, \alpha)}$$ NEGATIVE INTROSPECTION where f_{csf} and $f_{formulas}$ have been defined in the previous section, while $S_{STM}(i)$ (and its relatives, used later on) are defined as follows.

The set $S_{th}(i)$ is replaced by its memory-bank-specific counterparts, $S_{QTM}(i)$, $S_{new\text{-}STM}(i)$, $S_{STM}(i)$ and $S_{RTM}(i)$:

$$S_{theorems} \stackrel{df}{=} \{\boxed{(l,t,c,j,p,r) : \alpha} \in S_{du} | S_{axioms} \vdash \boxed{(l,t,c,j,p,r) : \alpha}\} \tag{4.7}$$

$$S_{QTM}(i) \stackrel{df}{=} \{\boxed{(l,t,c,j,p,r) : \alpha} \in S_{theorems} | (j = i) \wedge (l = QTM)\} \tag{4.8}$$

$$S_{STM}(i) \stackrel{df}{=} \{\boxed{(l,t,c,j,p,r) : \alpha} \in S_{theorems} | (j = i) \wedge (l = STM)\} \tag{4.9}$$

$$S_{new\text{-}STM}(i) \stackrel{df}{=} \{\boxed{(l,t,c,j,p,r) : \alpha} \in S_{STM}(i) | p = i\} \tag{4.10}$$

$$\begin{aligned} S_{RTM}(i) \stackrel{df}{=} \{&\boxed{(l,t,c,j,p,r) : \alpha} \in S_{theorems} | \\ &(j = i) \wedge ((l = STM) \vee ((l = ITM) \wedge (r > 0)))\} \end{aligned} \tag{4.11}$$

Just like $S_{th}(i)$, they are computable by purely syntactic operations and can be defined recursively on i.

The memory model in [DMP86] and step logic use different methods to detect and handle contradictions. Step logic indicates detected contradictions with the *Contra* predicate while the memory model instead uses the *loses* predicate. The memory model method uses certainty levels and the *loses* predicate is involved in the *RTM* mechanism. We have included both methods:

(CD1) $$\frac{\begin{array}{c}(STM, \varepsilon, c, i, p_1, R) : \alpha \\ (STM, \varepsilon, c, i, p_2, R) : \neg\alpha\end{array}}{(QTM, \varepsilon, C, i, 0, 0) : Contra(i, \alpha, \neg\alpha)}$$ Contradiction detection, same certainty

(CD2A) $$\frac{\begin{array}{l}(STM, \varepsilon, c_1, i, p_1, R) : \alpha \\ (STM, \varepsilon, c_2, i, p_2, R) : \neg\alpha \\ c_1 < c_2\end{array}}{(QTM, \varepsilon, C, i, 0, 0) : loses(\alpha)}$$ Contradiction detection, different certainties

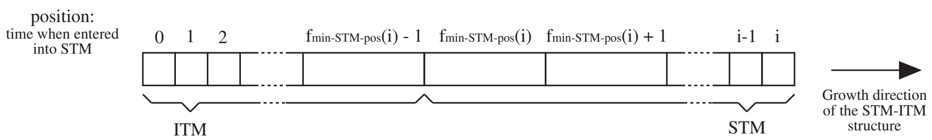
(CD2B) $$\frac{\begin{array}{l}(STM, \varepsilon, c_1, i, p_1, R) : \neg\alpha \\ (STM, \varepsilon, c_2, i, p_2, R) : \alpha \\ c_1 < c_2\end{array}}{(QTM, \varepsilon, C, i, 0, 0) : loses(\alpha)}$$ Contradiction detection, different certainties

The next group of rules handles inheritance, i.e., governs the time a particular formula stays in a memory bank or is moved to another one. The first inheritance rule says that everything in *LTM* stays in *LTM* forever:

$$\text{(IL)} \quad \frac{(LTM, \alpha, c, i, 0, 0) : \beta}{(LTM, \alpha, c, i+1, 0, 0) : \beta} \quad \textsc{Inheritance in LTM}$$

The STM is implemented as a FIFO queue of *sets* of declarative units rather than as a FIFO queue of declarative units. This "lazy" implementation avoids selection of a single formula among the QTM content because all of the QTM content can be shifted directly into the STM. But this has the disadvantage that it makes it a bit difficult to compare results based on different STM sizes in the LDS to similar results with the original memory model. Another problem is that the QTM content might become quite large if many formulae are retrieved in parallel from LTM by the SR rule.

The constant S is the STM size. We could have interpreted S as the maximum number of positions in STM and defined the STM content at time i as consisting of the last S positions $max(0, i - S + 1)..i$ of the structure. But that would have had the effect that formulae would "time out" from STM into ITM even when no new formulae were entered into STM. That is not the FIFO behavior described in [DMP86]. To get a real FIFO behavior we let STM contain a variable number of positions $f_{\textit{min-STM-pos}}(i)..i$ and instead interpret S as the maximum number of non-empty sets in the FIFO queue:



The function $f_{\textit{min-STM-pos}}$ is defined in terms of another function $f_{\textit{STM-pos}}$:

$$f_{\textit{min-STM-pos}}(i) \stackrel{df}{=} f_{\textit{STM-pos}}(i, i, S) \tag{4.12}$$

$f_{\textit{STM-pos}} : \mathbb{N}^3 \to \mathbb{N}$ is a recursive function computing the position of the sth nonempty subset counting backwards from position p at time i:

$$f_{\textit{STM-pos}}(i, p, s) \stackrel{df}{=}$$

$$\stackrel{df}{=} \begin{cases} p & \text{if } ((p = 0) \vee ((s = 1) \wedge \\ & \wedge (\exists \boxed{(l, \alpha, c, i', p', r) : \beta} \in S_{STM}(i))(p' = p)) \\ f_{\textit{STM-pos}}(i, p-1, s-1) & \text{if } (\exists \boxed{(l, \alpha, c, i', p', r) : \beta} \in S_{STM}(i))(p' = p) \\ f_{\textit{STM-pos}}(i, p-1, s) & \text{otherwise} \end{cases} \tag{4.13}$$

One problem with the "lazy" STM implementation is that limiting the number of non-empty sets in the STM does not necessarily limit the number of formulae in the STM. Many formulae can be moved into STM at the same time. Many elements in the STM sets means more computation per inference step. The flow from QTM to STM must be controlled to limit the amount of computation to realistic levels.

Useful formulae from QTM are promoted to STM. Because of the "lazy" STM implementation with sets of formulae in each position instead of single formulae we do not have to do much selection here. We just want to avoid multiple copies of the same formula in STM. We also make use of the RTM content to avoid rework on contradictions which have already been resolved:

$$\text{(IQS)}\quad \frac{\begin{array}{l}(QTM,\varepsilon,c,i,0,0):\alpha\\ \alpha\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \mathit{loses}(\alpha)\notin f_{\mathit{formulas}}(S_{RTM}(i))\end{array}}{(STM,\varepsilon,c,i+1,i+1,R):\alpha}\quad \textsc{Inheritance QTM}\rightarrow\text{STM}$$

When new formulae are entered into STM from QTM, old formulae must be pushed out of STM into ITM, to get a FIFO behaviour and to limit the STM size to S. This is done by the (IS) and (ISI) rules which use the function $f_{\mathit{min\text{-}STM\text{-}pos}}$ defined above:

$$\text{(IS)}\quad \frac{\begin{array}{l}(STM,\varepsilon,c,i,p,R):\alpha\\ (p>f_{\mathit{min\text{-}STM\text{-}pos}}(i))\vee(S_{\mathit{new\text{-}STM}}(i+1)=\emptyset)\\ \mathit{Contra}(i-1,\alpha,\beta)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \mathit{Contra}(i-1,\beta,\alpha)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \mathit{loses}(\alpha)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \alpha\neq \mathit{Now}(i)\\ (\alpha\neq K(i-1,\beta))\vee(K(i,\beta)\notin S_{QTM}(i))\\ (\alpha\neq \mathit{Contra}(i-1,\beta,\gamma))\vee(\mathit{Contra}(i,\beta,\gamma)\notin S_{QTM}(i))\end{array}}{(STM,\varepsilon,c,i+1,p,R):\alpha}\quad \textsc{Inheritance in STM}$$

$$\text{(ISI)}\quad \frac{\begin{array}{l}(STM,\varepsilon,c,i,p,R):\alpha\\ (p=f_{\mathit{min\text{-}STM\text{-}pos}}(i))\wedge(S_{\mathit{new\text{-}STM}}(i+1)\neq\emptyset)\\ \mathit{Contra}(i-1,\alpha,\beta)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \mathit{Contra}(i-1,\beta,\alpha)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ \mathit{loses}(\alpha)\notin f_{\mathit{formulas}}(S_{STM}(i))\\ (\alpha\neq K(i-1,\beta))\vee(K(i,\beta)\notin S_{QTM}(i))\\ (\alpha\neq \mathit{Contra}(i-1,\beta,\gamma))\vee(\mathit{Contra}(i,\beta,\gamma)\notin S_{QTM}(i))\end{array}}{(ITM,\varepsilon,c,i+1,p,R):\alpha}\quad \textsc{Inheritance STM}\rightarrow\text{ITM}$$

$$\text{(II)}\quad \frac{(ITM,\varepsilon,c,i,p,r):\alpha}{(ITM,\varepsilon,c,i+1,p,max(0,r-1)):\alpha}\quad \textsc{Inheritance in ITM}$$

We can now define the LDS which is intended to be a formalization of the memory model:

Definition 4.16 (Memory model LDS)
$\mathbb{L}_{mm} \stackrel{df}{=} (S_{labels}, \mathbf{L}, \mathbb{R}_{mm})$, where $\mathbb{R}_{mm}$ is built around (SR), (MP), (EMP), (NI), (CD1), (CD2A), (CD2B), (IL), (IQS), (IS), (ISI) and (II).

S_{labels} is interpreted as an appropriate algebra.

Appendix A contains two examples of simple default reasoning in $\mathbb{L}_{mm}$. They are both based on the default reasoning example used in [DMP86]. In the first example, a contradiction between formulae at the same certainty level is handled by using the *Contra* predicate. In the second example, a contradicition between formulae at different certainty levels is handled by using the *loses* predicate and the RTM mechanism.

Because we have no formal definition of the memory model in [DMP86], we can not prove the correctness of $\mathbb{L}_{mm}$ in the same way as we proved it for $\mathbb{L}_{SL_7}$. The examples above give some credibility to $\mathbb{L}_{mm}$ but it should be implemented and tested thoroughly, as suggested in Section 5.3.

4.5.4 Eliminating the references outside the database

One problem with $\mathbb{L}_{SL_7}$ and $\mathbb{L}_{mm}$ is that the functions $S_{th}(i)$, $S_{QTM}(i)$, $S_{STM}(i)$, $S_{new\text{-}STM}(i)$ and $S_{RTM}(i)$ refer to subsets of $S_{theorems}$ which have an uncertain relation to the contents of the current database. The sets are certainly computable, because one can compute them by starting from the axioms and apply the inference rules to all possible combinations of declarative units for i time steps. But since one does not know if these sets are contained in the current database, the functions can be said to refer to the results of "other proof processes". This is a bit ugly and is outside Gabbay's general pattern for inference rules for algebraic LDS.

The sets mentioned above are contained in the current database if the proof process is "completed" up to level i. In an implementation the time proceeds step by step and at each step the inference rules are applied to every possible combination of declarative units. So the "complete" sets above automatically become parts of the current database. But when describing the system as an algebraic LDS we can not be sure of the "completeness level" of an arbitrary database. The requirement for completeness demands restrictions on the order in which inference rules are applied; some of the rules can not be applied to some of the declarative units until the database has reached a certain level of completeness. This ordering is enforced by using the functions above.

One strength of LDS is that it can handle various features which are normally metalevel at the object level. It turns out that it can handle this ordering of the application of inference rules too. One way to extend $\mathbb{L}_{SL_7}$ to do that is described below.

In $\mathbb{L}_{SL_7}$ we only have the $S_{th}(i)$ sets to worry about. What we want is some way to keep track of the completeness level of the database. This can be done by extending the labels. Let us define

$$S_{rules} \stackrel{df}{=} \{I1, I2, I3, I4, I5\} \tag{4.14}$$

and

$$S_{comb} \stackrel{df}{=} S_{rules} \times \bigcup_{n \in \mathbb{Z}_+} (S_{du} \cup S_{wff})^n \tag{4.15}$$

S_{comb} is the set of all possible combinations of inference rules and tuples of formulae and declarative units.

Let us now introduce the function f_{comb} which computes the set of all valid applications of the inference rules to a set of declarative units:

$$f_{comb} : \mathcal{P}(S_{du}) \rightarrow \mathcal{P}(S_{comb}) \tag{4.16}$$

S_{labels} needs to be extended to $\mathbb{N} \cup (\mathbb{N} \times \mathcal{P}(S_{comb}))$. A label can be

- a point in time $i \in \mathbb{N}$;
- a tuple $(l, S_{remaining})$ where $l \in \mathbb{N}$ is the completeness level of the current database and $S_{remaining} \subset S_{comb}$ is the set of the applications of the inference rules which remain to be done before reaching $l+1$-completeness.

We can now formulate requirements on what constitutes a valid database:

- One declarative unit $\boxed{(l, S_{remaining}) : \mathit{level}}$ is present in the database.
- No more declarative units using the predicate *level* are present in the database.

The next step is to introduce $\boxed{(0, \emptyset) : \mathit{level}}$ as an axiom.

Let us define

$$S_{ax}(i) \stackrel{df}{=} \{\text{all axioms with label } i\} \tag{4.17}$$

and

$$S_{dus}(\Delta, i) \stackrel{df}{=} \{\text{all declarative units with label } i \text{ in the database } \Delta\} \tag{4.18}$$

Now we add a new rule I6 for level switching:

$$\text{(I6)} \quad \frac{(l, \emptyset) : \mathit{level}}{(l+1, f_{comb}(S_{dus}(\Delta, l) \cup S_{ax}(l+1))) : \mathit{level}} \quad \begin{array}{c}\text{LEVEL}\\ \text{SWITCH}\end{array}$$

where Δ refers to the current database.

The first application of rule I6 will generate the declarative unit $\boxed{(1, f_{comb}(S_{ax}(1))) : \mathit{level}}$.

Finally we need to rewrite the existing inference rules in order to make sure that a proper update is made to the *level* declarative unit:

$$\text{(I1)} \quad \frac{(i, S_{remaining}) : \mathit{level}, i : \alpha, i : \alpha \to \beta}{\begin{array}{l} i+1 : \beta \\ (i, S_{remaining} - \{(\mathit{I1}, i : \alpha, i : \alpha \to \beta)\}) : \mathit{level} \end{array}} \quad \textsc{Modus Ponens}$$

$$\text{(I2)} \quad \frac{\begin{array}{l} (i, S_{remaining}) : \mathit{level} \\ i : P_1 a \\ \ldots \\ i : P_n a \\ i : (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \to Qx] \end{array}}{\begin{array}{l} i+1 : Qa \\ (i, S_{remaining} - \{(\mathit{I2}, i : P_1 a, \ldots, i : P_n a \\ i : (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \to Qx])\}) : \mathit{level} \end{array}} \quad \textsc{Extended Modus Ponens}$$

$$\text{(I3)} \quad \frac{\begin{array}{l} (i, S_{remaining}) : \mathit{level} \\ \alpha \in f_{csf}(S_{dus}(\Delta, i)) \\ \alpha \notin f_{formulas}(S_{dus}(\Delta, i)) \end{array}}{i+1 : \neg K(i, \alpha), (i, S_{remaining} - \{(\mathit{I3}, \alpha)\}) : \mathit{level}} \quad \textsc{Negative Introspection}$$

$$\text{(I4)} \quad \frac{(i, S_{remaining}) : \mathit{level}, i : \alpha, i : \neg\alpha}{\begin{array}{l} i+1 : \mathit{Contra}(i, \alpha, \neg\alpha) \\ (i, S_{remaining} - \{(\mathit{I4}, i : \alpha)\}) : \mathit{level} \end{array}} \quad \textsc{Contradiction Detection}$$

$$\text{(I5)} \quad \frac{\begin{array}{l} (i, S_{remaining}) : \mathit{level} \\ i : \alpha \\ \mathit{Contra}(i-1, \alpha, \beta) \notin f_{formulas}(S_{dus}(\Delta, i)) \\ \mathit{Contra}(i-1, \beta, \alpha) \notin f_{formulas}(S_{dus}(\Delta, i)) \\ \alpha \neq \mathit{Now}(i) \end{array}}{i+1 : \alpha, (i, S_{remaining} - \{(\mathit{I5}, i : \alpha)\}) : \mathit{level}} \quad \textsc{Inheritance}$$

The references to $S_{th}(i)$ can be safely replaced with $S_{dus}(\Delta, i)$, because the i-completeness requirement in each of the rules I1-I5 guarantees that $S_{dus}(\Delta, i) = S_{th}(i)$.

The references outside the current database in $\mathbb{L}_{mm}$ can be removed in a similar manner. The ordering of the applications of inference rules has to be even more controlled here. Ordering between time steps is not enough. The dependencies between the subsets of the memory banks: $S_{QTM}(i)$, $S_{STM}(i)$, $S_{new\text{-}STM}(i)$ and $S_{RTM}(i)$ caused by the inference rules define a topological order of "substeps" of each time step. The correct ordering of these substeps can be guaranteed by defining several levels of completeness within each time step.

Chapter 5

Conclusions

5.1 Logical reasoning in general

The following is what I found out in my survey of existing methods:

First there are some unescapable theoretical limits on what can be achieved. As described in Section 2.1 there is a trade-off between the expressiveness of the logical language and the complexity of the associated computational problems. Polynomially tractable logics exist and the tractability threshold between these and other logics has been studied.

The capturing results from descriptive complexity provide time limits for model checking but the relation between these limits and the time limits for syntactical operations is non-obvious. The capturing results are connected with central questions in complexity theory, such as whether $\mathbf{P} = \mathbf{NP}$ and *GRAPH-ISOMORPHISM* $\in \mathbf{P}$.

A lot of methods for efficient logical reasoning have been described in the literature. First, the complexity of the computational problems can be reduced by making a good choice of language (the expressiveness-complexity trade-off above). Language restriction is described in Section 3.2. And as described in Section 3.3.2, it is sometimes possible to adjust the strength of the language by using different semantics for the same syntax.

When the choice of language and semantics has been done and the computational problem is thus fixed, the average or typical case can be improved by using good inference rules and strategies as described in Section 3.4.

Using theory approximation as described in Section 3.5 can give an indication of the hardness of a reasoning problem before the real answer is found.

Knowledge compilation which is described in Section 3.6 can reduce the total amount of computation when querying a knowledge base where updates are rare. Knowledge compilation can also be used to move the computational effort in time. Most of the computation can be done in advance, which might improve response time for critical queries.

Active logic described in Chapter 4 is a family of logics with some unique abilities. The reasoning is situated in time and it is possible to do metareasoning about the reasoning process itself. Active logic can handle contradictions and it solves the omniscience problem. It also makes efficient nonmonotonic reasoning possible.

The computational requirements of step logics are rather bad. But the original memory model, which can be expressed with $\mathbb{L}_{mm}$ as shown in this report, and the modal variants of active logic in [Nir94] and [Glo97] have nicer computational properties. One interesting property of active logic is the possibility for efficient nonmonotonic reasoning.

Deliberation scheduling can be seen as metareasoning which can be applied on top of other methods. One combination which could work is with improvable theory approximation where the sequence of approximations provide the possibility to explicitly allocate computational resources. A combination with active logic could also work. The possibility in active logic to reason about the reasoning process means that the scheduling itself could be implemented in active logic.

5.2 Active logic and LDS

The following is what I found out in my attempt to describe active logic as an LDS.

I have shown that LDS methodology can be applied to active logics by giving two simple examples, $\mathbb{L}_{SL_7}$ and $\mathbb{L}_{mm}$. Some of the inference rules in these systems are outside Gabbay's standard types of inference rules for algebraic LDS. One interesting question is if this can be avoided. If that is not possible, the effect of the non-standard rules on proof theory and semantics remains to be investigated, see below.

Describing the memory model "directly" in $\mathbb{L}_{mm}$ results in a more complete formalization than step logics and one which has nicer computational properties.

5.3 Further research

Here are some suggestions for further research:

- $\mathbb{L}_{mm}$, the memory model LDS, should be implemented. Testing an implementation with realistically large problems may tell us more about the properties of the system than just simulating it by hand.

- The proof theory and semantics of $\mathbb{L}_{SL_7}$ and $\mathbb{L}_{mm}$ should be investigated. If the new types of rules outside the standard types of rules in [Gab96] can not be avoided, that might be a lot of work.

- The "lazy" STM implementation in $\mathbb{L}_{mm}$ using a FIFO queue of sets of declarative units can be replaced with a real one using a FIFO queue of declarative units. This requires selection among the QTM contents.

- Our mechanism with certainty levels is rather simple, but [Gab96] describes how truth maintenance systems can be implemented in LDS by storing proofs in labels. So there is hope for a better solution.

- The $\mathbb{L}_{mm}$ system does not include episodic retrieval from ITM. [DMP86] does not describe episodic retrieval in detail but says that it can be useful when implementing a goal-subgoal process to achieve goal-directed behavior.
- The LTM contents is static. The possibility of augmenting it (e.g., by some learning process) could be investigated.
- The modal variants of active logic from [Nir94] and [Glo97] should be expressed as LDS. LDS works well for more conventional modal logics and seems to work with active logics in my $\mathbb{L}_{SL_7}$ and $\mathbb{L}_{mm}$ systems. So it seems reasonable to believe that it should work with modal active logics as well.

Appendix A

Some proofs in $\mathbb{L}_{mm}$

Contradiction handling using the *Contra* predicate:

d.u. number	declarative unit	premisses
1	$(LTM, bird(x), C, 0, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	axiom
2	$(LTM, ostrich(x), C, 0, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	axiom
3	$(STM, \varepsilon, C, 0, 0, 20) : Now(0)$	axiom
4	$(QTM, \varepsilon, C, 0, 0, 0) : bird(Tweety)$	axiom
5	$(LTM, bird(x), C, 1, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 1
6	$(LTM, ostrich(x), C, 1, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 2
7	$(STM, \varepsilon, C, 1, 1, 20) : Now(1)$	axiom
8	$(STM, \varepsilon, C, 1, 1, 20) : bird(Tweety)$	IQS 4
9	$(QTM, \varepsilon, C, 1, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 8,5
10	$(LTM, bird(x), C, 2, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 5
11	$(LTM, ostrich(x), C, 2, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 6
12	$(STM, \varepsilon, C, 2, 1, 20) : bird(Tweety)$	IS 8
13	$(STM, \varepsilon, C, 2, 2, 20) : Now(2)$	axiom
14	$(STM, \varepsilon, C, 2, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IQS 9
15	$(QTM, \varepsilon, C, 2, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 12,10
16	$(QTM, \varepsilon, C, 2, 0, 0) : \neg K(2, \neg flies(Tweety))$	NI 14
17	$(QTM, \varepsilon, C, 2, 0, 0) : \neg K(2, flies(Tweety))$	NI 14

d.u. number	declarative unit	premisses
18	$(LTM, bird(x), C, 3, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 10
19	$(LTM, ostrich(x), C, 3, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 11
20	$(STM, \varepsilon, C, 3, 1, 20) : bird(Tweety)$	IS 12
21	$(STM, \varepsilon, C, 3, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 14
22	$(STM, \varepsilon, C, 3, 3, 20) : Now(3)$	axiom
23	$(STM, \varepsilon, C, 3, 3, 20) : \neg K(2, \neg flies(Tweety))$	IQS 16
24	$(STM, \varepsilon, C, 3, 3, 20) : \neg K(2, flies(Tweety))$	IQS 17
25	$(QTM, \varepsilon, C, 3, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 20,18
26	$(QTM, \varepsilon, C, 3, 0, 0) : \neg K(3, \neg flies(Tweety))$	NI 21
27	$(QTM, \varepsilon, C, 3, 0, 0) : \neg K(3, flies(Tweety))$	NI 21
28	$(QTM, \varepsilon, C, 3, 0, 0) : flies(Tweety)$	EMP 20,22,23,21
29	$(LTM, bird(x), C, 4, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 18
30	$(LTM, ostrich(x), C, 4, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 19
31	$(STM, \varepsilon, C, 4, 1, 20) : bird(Tweety)$	IS 20
32	$(STM, \varepsilon, C, 4, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 21
33	$(STM, \varepsilon, C, 4, 4, 20) : Now(4)$	axiom
34	$(STM, \varepsilon, C, 4, 4, 20) : \neg K(3, \neg flies(Tweety))$	IQS 26
35	$(STM, \varepsilon, C, 4, 4, 20) : \neg K(3, flies(Tweety))$	IQS 27
36	$(STM, \varepsilon, C, 4, 4, 20) : flies(Tweety)$	IQS 28
37	$(QTM, \varepsilon, C, 4, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 31,29
38	$(QTM, \varepsilon, C, 4, 0, 0) : \neg K(4, \neg flies(Tweety))$	NI 32
39	$(QTM, \varepsilon, C, 4, 0, 0) : flies(Tweety)$	EMP 31,33,34,32
40	$(QTM, \varepsilon, C, 4, 0, 0) : ostrich(Tweety)$	axiom
41	$(LTM, bird(x), C, 5, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 29
42	$(LTM, ostrich(x), C, 5, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 30
43	$(STM, \varepsilon, C, 5, 1, 20) : bird(Tweety)$	IS 31
44	$(STM, \varepsilon, C, 5, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 32
45	$(STM, \varepsilon, C, 5, 4, 20) : \neg K(3, flies(Tweety))$	IS 35
46	$(STM, \varepsilon, C, 5, 4, 20) : flies(Tweety)$	IS 36
47	$(STM, \varepsilon, C, 5, 5, 20) : Now(5)$	axiom
48	$(STM, \varepsilon, C, 5, 5, 20) : \neg K(4, \neg flies(Tweety))$	IQS 38
49	$(STM, \varepsilon, C, 5, 5, 20) : ostrich(Tweety)$	IQS 40
50	$(QTM, \varepsilon, C, 5, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 43,41
51	$(QTM, \varepsilon, C, 5, 0, 0) : \neg K(5, \neg flies(Tweety))$	NI 44
52	$(QTM, \varepsilon, C, 5, 0, 0) : flies(Tweety)$	EMP 43,47,48,44
53	$(QTM, \varepsilon, C, 5, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 49,42

d.u. number	declarative unit	premisses
54	$(LTM, bird(x), C, 6, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 41
55	$(LTM, ostrich(x), C, 6, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 42
56	$(STM, \varepsilon, C, 6, 1, 20) : bird(Tweety)$	IS 43
57	$(STM, \varepsilon, C, 6, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 44
58	$(STM, \varepsilon, C, 6, 4, 20) : \neg K(3, flies(Tweety))$	IS 45
59	$(STM, \varepsilon, C, 6, 4, 20) : flies(Tweety)$	IS 46
60	$(STM, \varepsilon, C, 6, 5, 20) : ostrich(Tweety)$	IS 49
61	$(STM, \varepsilon, C, 6, 6, 20) : Now(6)$	axiom
62	$(STM, \varepsilon, C, 6, 6, 20) : \neg K(5, \neg flies(Tweety))$	IQS 51
63	$(STM, \varepsilon, C, 6, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IQS 53
64	$(QTM, \varepsilon, C, 6, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 56,54
65	$(QTM, \varepsilon, C, 6, 0, 0) : \neg K(6, \neg flies(Tweety))$	NI 57
66	$(QTM, \varepsilon, C, 6, 0, 0) : flies(Tweety)$	EMP 56,61,62,57
67	$(QTM, \varepsilon, C, 6, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 60,55
68	$(QTM, \varepsilon, C, 6, 0, 0) : \neg flies(Tweety)$	MP 60,63
69	$(LTM, bird(x), C, 7, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 54
70	$(LTM, ostrich(x), C, 7, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 55
71	$(STM, \varepsilon, C, 7, 1, 20) : bird(Tweety)$	IS 56
72	$(STM, \varepsilon, C, 7, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 57
73	$(STM, \varepsilon, C, 7, 4, 20) : \neg K(3, flies(Tweety))$	IS 58
74	$(STM, \varepsilon, C, 7, 4, 20) : flies(Tweety)$	IS 59
75	$(STM, \varepsilon, C, 7, 5, 20) : ostrich(Tweety)$	IS 60
76	$(STM, \varepsilon, C, 7, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IS 63
77	$(STM, \varepsilon, C, 7, 7, 20) : Now(7)$	axiom
78	$(STM, \varepsilon, C, 7, 7, 20) : \neg K(6, \neg flies(Tweety))$	IQS 65
79	$(STM, \varepsilon, C, 7, 7, 20) : \neg flies(Tweety)$	IQS 68
80	$(QTM, \varepsilon, C, 7, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 71,69
81	$(QTM, \varepsilon, C, 7, 0, 0) : flies(Tweety)$	EMP 71,77,78,72
82	$(QTM, \varepsilon, C, 7, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 75,70
83	$(QTM, \varepsilon, C, 7, 0, 0) : \neg flies(Tweety)$	MP 75,76
84	$(QTM, \varepsilon, C, 7, 0, 0) : Contra(7, flies(Tweety),$ $\neg flies(Tweety))$	CD1 74,79

d.u. number	declarative unit	premisses
85	$(LTM, bird(x), C, 8, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \to flies(x)]$	IL 69
86	$(LTM, ostrich(x), C, 8, 0, 0) : (\forall x)[ostrich(x) \to \neg flies(x)]$	IL 70
87	$(STM, \varepsilon, C, 8, 1, 20) : bird(Tweety)$	IS 71
88	$(STM, \varepsilon, C, 8, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge \neg K(i-1, \neg flies(Tweety))) \to flies(Tweety)]$	IS 72
89	$(STM, \varepsilon, C, 8, 4, 20) : \neg K(3, flies(Tweety))$	IS 73
90	$(STM, \varepsilon, C, 8, 4, 20) : flies(Tweety)$	IS 74
91	$(STM, \varepsilon, C, 8, 5, 20) : ostrich(Tweety)$	IS 75
92	$(STM, \varepsilon, C, 8, 6, 20) : ostrich(Tweety) \to \neg flies(Tweety)$	IS 76
93	$(STM, \varepsilon, C, 8, 7, 20) : \neg K(6, \neg flies(Tweety))$	IS 78
94	$(STM, \varepsilon, C, 8, 7, 20) : \neg flies(Tweety)$	IS 79
95	$(STM, \varepsilon, C, 8, 8, 20) : Now(8)$	axiom
96	$(STM, \varepsilon, C, 8, 8, 20) : Contra(7, flies(Tweety), \neg flies(Tweety))$	IQS 84
97	$(QTM, \varepsilon, C, 8, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge \neg K(i-1, \neg flies(Tweety))) \to flies(Tweety)]$	SR 87,85
98	$(QTM, \varepsilon, C, 8, 0, 0) : ostrich(Tweety) \to \neg flies(Tweety)$	SR 91,86
99	$(QTM, \varepsilon, C, 8, 0, 0) : \neg flies(Tweety)$	MP 91,92
100	$(QTM, \varepsilon, C, 8, 0, 0) : Contra(8, flies(Tweety), \neg flies(Tweety))$	CD1 90,94
101	$(LTM, bird(x), C, 9, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge \neg K(i-1, \neg flies(x))) \to flies(x)]$	IL 85
102	$(LTM, ostrich(x), C, 9, 0, 0) : (\forall x)[ostrich(x) \to \neg flies(x)]$	IL 86
103	$(STM, \varepsilon, C, 9, 1, 20) : bird(Tweety)$	IS 87
104	$(STM, \varepsilon, C, 9, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge \neg K(i-1, \neg flies(Tweety))) \to flies(Tweety)]$	IS 88
105	$(STM, \varepsilon, C, 9, 4, 20) : \neg K(3, flies(Tweety))$	IS 89
106	$(STM, \varepsilon, C, 9, 5, 20) : ostrich(Tweety)$	IS 91
107	$(STM, \varepsilon, C, 9, 6, 20) : ostrich(Tweety) \to \neg flies(Tweety)$	IS 92
108	$(STM, \varepsilon, C, 9, 7, 20) : \neg K(6, \neg flies(Tweety))$	IS 93
109	$(STM, \varepsilon, C, 9, 9, 20) : Now(9)$	axiom
110	$(STM, \varepsilon, C, 9, 9, 20) : Contra(8, flies(Tweety), \neg flies(Tweety))$	IQS 100
111	$(QTM, \varepsilon, C, 9, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge \neg K(i-1, \neg flies(Tweety))) \to flies(Tweety)]$	SR 103,101
112	$(QTM, \varepsilon, C, 9, 0, 0) : ostrich(Tweety) \to \neg flies(Tweety)$	SR 106,102
113	$(QTM, \varepsilon, C, 9, 0, 0) : \neg flies(Tweety)$	MP 106,107
114	$(QTM, \varepsilon, C, 9, 0, 0) : \neg K(9, \neg flies(Tweety))$	NI 104
115	$(QTM, \varepsilon, C, 9, 0, 0) : \neg K(9, flies(Tweety))$	NI 104

Contradiction handling using the *loses* predicate and the RTM mechanism:

d.u. number	declarative unit	premisses
1	$(LTM, bird(x), U, 0, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	axiom
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	axiom
2	$(LTM, ostrich(x), C, 0, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	axiom
3	$(STM, \varepsilon, C, 0, 0, 20) : Now(0)$	axiom
4	$(QTM, \varepsilon, C, 0, 0, 0) : bird(Tweety)$	axiom
5	$(LTM, bird(x), U, 1, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	IL 1
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	
6	$(LTM, ostrich(x), C, 1, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 2
7	$(STM, \varepsilon, C, 1, 1, 20) : Now(1)$	axiom
8	$(STM, \varepsilon, C, 1, 1, 20) : bird(Tweety)$	IQS 4
9	$(QTM, \varepsilon, U, 1, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	SR 8,5
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
10	$(LTM, bird(x), U, 2, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	IL 5
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	
11	$(LTM, ostrich(x), C, 2, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 6
12	$(STM, \varepsilon, C, 2, 1, 20) : bird(Tweety)$	IS 8
13	$(STM, \varepsilon, C, 2, 2, 20) : Now(2)$	axiom
14	$(STM, \varepsilon, U, 2, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	IQS 9
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
15	$(QTM, \varepsilon, U, 2, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	SR 12,10
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
16	$(QTM, \varepsilon, C, 2, 0, 0) : \neg K(2, \neg flies(Tweety))$	NI 14
17	$(QTM, \varepsilon, C, 2, 0, 0) : \neg K(2, flies(Tweety))$	NI 14
18	$(LTM, bird(x), U, 3, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	IL 10
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	
19	$(LTM, ostrich(x), C, 3, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 11
20	$(STM, \varepsilon, C, 3, 1, 20) : bird(Tweety)$	IS 12
21	$(STM, \varepsilon, U, 3, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	IS 14
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
22	$(STM, \varepsilon, C, 3, 3, 20) : Now(3)$	axiom
23	$(STM, \varepsilon, C, 3, 3, 20) : \neg K(2, \neg flies(Tweety))$	IQS 16
24	$(STM, \varepsilon, C, 3, 3, 20) : \neg K(2, flies(Tweety))$	IQS 17
25	$(QTM, \varepsilon, U, 3, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	SR 20,18
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
26	$(QTM, \varepsilon, C, 3, 0, 0) : \neg K(3, \neg flies(Tweety))$	NI 21
27	$(QTM, \varepsilon, C, 3, 0, 0) : \neg K(3, flies(Tweety))$	NI 21
28	$(QTM, \varepsilon, U, 3, 0, 0) : flies(Tweety)$	EMP 20,22,23,21

d.u. number	declarative unit	premisses
29	$(LTM, bird(x), U, 4, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 18
30	$(LTM, ostrich(x), C, 4, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 19
31	$(STM, \varepsilon, C, 4, 1, 20) : bird(Tweety)$	IS 20
32	$(STM, \varepsilon, U, 4, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 21
33	$(STM, \varepsilon, C, 4, 4, 20) : Now(4)$	axiom
34	$(STM, \varepsilon, C, 4, 4, 20) : \neg K(3, \neg flies(Tweety))$	IQS 26
35	$(STM, \varepsilon, C, 4, 4, 20) : \neg K(3, flies(Tweety))$	IQS 27
36	$(STM, \varepsilon, U, 4, 4, 20) : flies(Tweety)$	IQS 28
37	$(QTM, \varepsilon, U, 4, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 31,29
38	$(QTM, \varepsilon, C, 4, 0, 0) : \neg K(4, \neg flies(Tweety))$	NI 32
39	$(QTM, \varepsilon, U, 4, 0, 0) : flies(Tweety)$	EMP 31,33,34,32
40	$(QTM, \varepsilon, C, 4, 0, 0) : ostrich(Tweety)$	axiom
41	$(LTM, bird(x), U, 5, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 29
42	$(LTM, ostrich(x), C, 5, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 30
43	$(STM, \varepsilon, C, 5, 1, 20) : bird(Tweety)$	IS 31
44	$(STM, \varepsilon, U, 5, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 32
45	$(STM, \varepsilon, U, 5, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	IS 35
46	$(STM, \varepsilon, U, 5, 4, 20) : flies(Tweety)$	IS 36
47	$(STM, \varepsilon, C, 5, 5, 20) : Now(5)$	axiom
48	$(STM, \varepsilon, C, 5, 5, 20) : \neg K(4, \neg flies(Tweety))$	IQS 38
49	$(STM, \varepsilon, C, 5, 5, 20) : ostrich(Tweety)$	IQS 42
50	$(QTM, \varepsilon, U, 5, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 43,41
51	$(QTM, \varepsilon, C, 5, 0, 0) : \neg K(5, \neg flies(Tweety))$	NI 44
52	$(QTM, \varepsilon, U, 5, 0, 0) : flies(Tweety)$	EMP 43,47,48,44
53	$(QTM, \varepsilon, C, 5, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 49,42

d.u. number	declarative unit	premisses
54	$(LTM, bird(x), U, 6, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 41
55	$(LTM, ostrich(x), C, 6, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 42
56	$(STM, \varepsilon, C, 6, 1, 20) : bird(Tweety)$	IS 43
57	$(STM, \varepsilon, U, 6, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 44
58	$(STM, \varepsilon, C, 6, 4, 20) : \neg K(3, flies(Tweety))$	IS 45
59	$(STM, \varepsilon, U, 6, 4, 20) : flies(Tweety)$	IS 46
60	$(STM, \varepsilon, C, 6, 5, 20) : ostrich(Tweety)$	IS 49
61	$(STM, \varepsilon, C, 6, 6, 20) : Now(6)$	axiom
62	$(STM, \varepsilon, C, 6, 6, 20) : \neg K(5, \neg flies(Tweety))$	IQS 51
63	$(STM, \varepsilon, C, 6, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IQS 53
64	$(QTM, \varepsilon, U, 6, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 56,54
65	$(QTM, \varepsilon, C, 6, 0, 0) : \neg K(6, \neg flies(Tweety))$	NI 57
66	$(QTM, \varepsilon, U, 6, 0, 0) : flies(Tweety)$	EMP 56,61,62,57
67	$(QTM, \varepsilon, C, 6, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 60,55
68	$(QTM, \varepsilon, C, 6, 0, 0) : \neg flies(Tweety)$	MP 60,63
69	$(LTM, bird(x), U, 7, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	IL 54
70	$(LTM, ostrich(x), C, 7, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 55
71	$(STM, \varepsilon, C, 7, 1, 20) : bird(Tweety)$	IS 56
72	$(STM, \varepsilon, U, 7, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 57
73	$(STM, \varepsilon, C, 7, 4, 20) : \neg K(3, flies(Tweety))$	IS 58
74	$(STM, \varepsilon, U, 7, 4, 20) : flies(Tweety)$	IS 59
75	$(STM, \varepsilon, C, 7, 5, 20) : ostrich(Tweety)$	IS 60
76	$(STM, \varepsilon, C, 7, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IS 63
77	$(STM, \varepsilon, C, 7, 7, 20) : Now(7)$	axiom
78	$(STM, \varepsilon, C, 7, 7, 20) : \neg K(6, \neg flies(Tweety))$	IQS 65
79	$(STM, \varepsilon, C, 7, 7, 20) : \neg flies(Tweety)$	IQS 68
80	$(QTM, \varepsilon, U, 7, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$ $\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	SR 71,69
81	$(QTM, \varepsilon, U, 7, 0, 0) : flies(Tweety)$	EMP 71,77,78,72
82	$(QTM, \varepsilon, C, 7, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 75,76
83	$(QTM, \varepsilon, C, 7, 0, 0) : \neg flies(Tweety)$	MP 75,76
84	$(QTM, \varepsilon, C, 7, 0, 0) : loses(flies(Tweety))$	CD2A 74,79

d.u. number	declarative unit	premisses
85	$(LTM, bird(x), U, 8, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	IL 69
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	
86	$(LTM, ostrich(x), C, 8, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 70
87	$(STM, \varepsilon, C, 8, 1, 20) : bird(Tweety)$	IS 71
88	$(STM, \varepsilon, U, 8, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	IS 72
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
89	$(STM, \varepsilon, C, 8, 4, 20) : \neg K(3, flies(Tweety))$	IS 73
90	$(STM, \varepsilon, U, 8, 4, 20) : flies(Tweety)$	IS 74
91	$(STM, \varepsilon, C, 8, 5, 20) : ostrich(Tweety)$	IS 75
92	$(STM, \varepsilon, C, 8, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IS 76
93	$(STM, \varepsilon, C, 8, 7, 20) : \neg K(6, \neg flies(Tweety))$	IS 78
94	$(STM, \varepsilon, C, 8, 7, 20) : \neg flies(Tweety)$	IS 79
95	$(STM, \varepsilon, C, 8, 8, 20) : Now(8)$	axiom
96	$(STM, \varepsilon, C, 8, 8, 20) : loses(flies(Tweety))$	IQS 84
97	$(QTM, \varepsilon, U, 8, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	SR 87,85
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
98	$(QTM, \varepsilon, C, 8, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 91,86
99	$(QTM, \varepsilon, C, 8, 0, 0) : \neg flies(Tweety)$	MP 91,92
100	$(QTM, \varepsilon, C, 8, 0, 0) : loses(flies(Tweety))$	CD2A 90,94
101	$(LTM, bird(x), U, 9, 0, 0) : (\forall x)(\forall i)[(bird(x) \wedge Now(i) \wedge$	IL 85
	$\neg K(i-1, \neg flies(x))) \rightarrow flies(x)]$	
102	$(LTM, ostrich(x), C, 9, 0, 0) : (\forall x)[ostrich(x) \rightarrow \neg flies(x)]$	IL 86
103	$(STM, \varepsilon, C, 9, 1, 20) : bird(Tweety)$	
104	$(STM, \varepsilon, U, 9, 2, 20) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	IS 87
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	IS 88
105	$(STM, \varepsilon, C, 9, 4, 20) : \neg K(3, flies(Tweety))$	IS 89
106	$(STM, \varepsilon, C, 9, 5, 20) : ostrich(Tweety)$	IS 91
107	$(STM, \varepsilon, C, 9, 6, 20) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	IS 92
108	$(STM, \varepsilon, C, 9, 7, 20) : \neg K(6, \neg flies(Tweety))$	IS 93
109	$(STM, \varepsilon, C, 9, 7, 20) : \neg flies(Tweety)$	IS 94
110	$(STM, \varepsilon, C, 9, 8, 20) : loses(flies(Tweety))$	IS 96
111	$(STM, \varepsilon, C, 9, 9, 20) : Now(9)$	axiom
112	$(QTM, \varepsilon, U, 9, 0, 0) : (\forall i)[(bird(Tweety) \wedge Now(i) \wedge$	SR 103,101
	$\neg K(i-1, \neg flies(Tweety))) \rightarrow flies(Tweety)]$	
113	$(QTM, \varepsilon, C, 9, 0, 0) : ostrich(Tweety) \rightarrow \neg flies(Tweety)$	SR 106,102
114	$(QTM, \varepsilon, C, 9, 0, 0) : \neg flies(Tweety)$	MP 106,107
115	$(QTM, \varepsilon, C, 9, 0, 0) : \neg K(9, flies(Tweety))$	NI 104

Bibliography

[BD94] M. Boddy and T. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67 (2):245–286, 1994.

[CD97] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[DG94] M. D'Agostino and D. Gabbay. A generalization of analytic deduction via labelled deductive systems. *Journal of Automated Reasoning*, 13:243–281, 1994.

[DMP86] Jennifer Drapkin, Michael Miller, and Donald Perlis. A memory model for real-time commonsense reasoning. Technical Report TR-86-21, Systems Research Center, University of Maryland, College Park, Maryland, 1986.

[Ebb99] H.-D. Ebbinghaus. Is there a logic for polynomial time? *Logic Journal of the IGPL*, 7(3):359–374, 1999.

[ED88] Jennifer Elgot-Drapkin. Step-logic: Reasoning situated in time. PhD thesis CS-TR-2156, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

[EDKM+99] Jennifer Elgot-Drapkin, Sarit Kraus, Michael Miller, Madhura Nirkhe, and Donald Perlis. Active logics: A unified formal approach to episodic reasoning. Technical Report CS-TR-4072, Department of Computer Science, University of Maryland, College Park, Maryland, 1999.

[EDMP91] Jennifer Elgot-Drapkin, Michael Miller, and Donald Perlis. Memory, reason, and time: the step-logic approach. In Cummins and Pollock, editors, *Philosophy and AI : Essays at the Interface*, chapter 4, pages 79–103. MIT Press, 1991.

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

[EF99] H. D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, second edition, 1999. ISBN 3-540-65758-4.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity and Computation*, volume 7, pages 43–73. SIAM-AMS Proceedings, 1974.

[Fri85] A. M. Frisch. Using model theory to specify AI programs. In *Proc. of the 9th IJCAI*, pages 148–154, Los Angeles, CA, 1985.

[Gab96] Dov M. Gabbay. *Labelled Deductive Systems*. Oxford University Press, 1996. ISBN 0-19-853833-2.

[Glo97] Ayelet Globerman. A modal active logic with focus of attention for reasoning in time. Master's thesis, Department of Mathematics and Computer Science, Bar Ilan University, Ramat Gan, Israel, 1997.

[GW01] D. Gabbay and J. Woods. The new logic. *Logic Journal of the IGPL*, 9(1):141–174, 2001.

[HCH89] Eric Horvitz, Gregory F. Cooper, and David Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proc 11th IJCAI*, pages 1121–1127, 1989.

[Lem65] E.J. Lemmon. *Beginning Logic*. Van Nostrand Reinhold (International) Co. Ltd, 11 New Fetter Lane, London EC4P 4EE, 1965. ISBN 0-442-30676-8.

[Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. of AAAI-84*, pages 198–202, Austin, TX, 1984.

[Nil91] Nils J. Nilsson. Logic and artificial intelligence. *Artificial Intelligence*, 47(1-3):31–56, 1991.

[Nir94] Madhura Nirkhe. *Time-Situated Reasoning within Tight Deadlines and Realistic Space and Computation Bounds*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1994.

[NKP93] M. Nirkhe, S. Kraus, and D. Perlis. Situated reasoning within tight deadlines and realistic space and computation bounds. In *Common Sense*, 1993.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. ISBN 0-201-53082-1.

[PS85] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. In *Proc. of the 9th IJCAI*, pages 455–458, Los Angeles, CA, 1985.

[PS86] P. F. Patel-Schneider. A four-valued semantics for frame-based description languages. In *Proc. of AAAI-86*, pages 344–348, Philadelphia, PA, 1986.

[RN95] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, New Jersey, 1995. ISBN 0-13-103805-2.

[Rob65] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[SC95] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

[SK96] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[Wos88] Larry Wos. *Automated Reasoning : 33 basic search problems*. Prentice Hall, New Jersey, 1988.

Index