Reasoning with Limited Resources: Active Logics Expressed as Labeled Deductive Systems

Mikael Asker, Jacek Malec*
Department of Computer Science
Lund University
Box 118
221 00 Lund, Sweden

Abstract

Reasoning with limited computational resources (such as time or memory) is an important problem, in particular in knowledge-intensive embedded systems. Classical logic is usually considered inappropriate for this purpose as no guarantees regarding deadlines can be made. One of the more interesting approaches to address this problem is built around the concept of *active logics*. Although a step in the right direction, active logics are just a preliminary attempt towards finding an acceptable solution.

Our work is based on the assumption that *Labeled Deductive Systems* offer appropriate metamathematical methodology to study the problem. As a first step, we have reformulated a pair of active logics systems, namely the *memory model* and its formalized simplification, the *step logic*, as Labeled Deductive Systems.

This paper presents our motivation behind this project, followed by an overview of the investigations on meta-reasoning relevant to this work, and introduces in some reasonable detail the MM system.

Keywords: Active Logic, Labeled Deductive Systems, reasoning with limited resources

*email: jacek@cs.lth.se

1 Introduction

Reasoning with limited computational resources (such as time or memory) is an important problem, in particular in knowledge-intensive embedded systems. Usually a decision, based on a principled reasoning process, needs to be taken within limited time and taking into account the limitations of the processing power and resources of the reasoning system. Therefore symbolic logic is often considered as an inadequate tool for implementing reasoning in such systems: logic does not guarantee that all relevant inferences will be made within prescribed time nor does it allow to limit the required memory resources satisfactorily. The paradigm shift that occurred in Artificial Intelligence in the middle of 1980s can be attributed to increasing awareness of those limitations inherently present in the then predominant way of representing and using knowledge.

Since then there have been some attempts to constrain the inference process performed in a logical system in a principled way. One possibility is to limit the expressive power of the first-order logical calculus (as, e.g., in description logics) in order to guarantee polynomial-time computability. Another is to use polynomial approximations of the reasoning process. Yet another is to constrain the inference process in order to retain control over it. More details about these lines of research can be found in Section 3.

One of the more interesting research in this area during 1990s has focused on logic as a model of an on-going reasoning process rather than as a static characterization of contents of a knowledge base. It begun with *step-logic* and evolved into a family of *active logics*. The most recent focus of this research is on modeling dialogue and discourse. However, other interesting applications, like planning or multi-agent systems, have also been investigated, while some other possibilities still await analysis. In particular, the possibility of applying this framework to resource-bounded reasoning in embedded systems is in the focus of our interest.

Finally, one should name the relations to the large area of *belief revision* that also investigates the process of knowledge update rather than the static aspects of logical theories. However, there has been little attention paid to possibilities of using this approach in resource-bounded reasoning - the work has rather focused on the pure non-monotonicity aspect of knowledge revision process.

The rest of the paper is divided as follows. Section 2 presents the background of the idea leading to our investigation. In Section 3 we discuss the relevant related work. Section 4 introduces the memory model being the foundation for active logics research. Then Section 5 presents an LDS formalization of the memory model. Section 6 discusses how the described

approach could be used for planning in real-time for robotic applications. Finally the conclusions and some suggestion of further work are presented.

2 Background

The very first idea for this investigation has been born from the naive hypothesis that in order to be able to use symbolic logical reasoning in a real-time system context it would be sufficient to limit the depth of reasoning to a given, predefined level. This way one would be able to guarantee predictability of a system using this particular approach to reasoning. Unfortunately, such a modification performed on a classical logical system yields a formalism with a heavily modified and in principle unknown semantics [1]. It would be necessary to relate it to the classical one in a thorough manner. This task seems very hard and it is unclear for us what techniques should be used to proceed along this line. But the very basic idea of "modified provability": *A formula is a theorem iff it is provable within n steps of reasoning*, is still appealing and will guide us in our investigations presented below.

The next observation made in the beginning of this work was that predictability (in the hard real-time sense) requires very tight control over the reasoning process. In the classical approach one specifies a number of axioms and a set of inference rules, and the entailed consequences are expected to "automagically" appear as results of an appropriate consequence relation. Unfortunately, this relation is very hard to compute and usually requires exponential algorithms. One possibility is to modify the consequence relation in such way that it becomes computable. However, the exact way of achieving that is far from obvious. We have investigated previous approaches (listed in Section 3) and concluded that a reasonable technique for doing this would be to introduce a mechanism that would allow one to control the inference process. One such mechanism is available in Labeled Deductive Systems [2].

In its most simple, somewhat trivialized setting a labeled deductive system (LDS) attaches a *label* to every well-formed formula and allows the inference rules to analyze and modify labels, or even trigger on specific conditions defined on the labels. E.g., instead of the classical Modus Ponens rule $\frac{A, A \rightarrow B}{B}$ a labeled deduction system would use $\frac{\alpha:A,\ \beta:A \rightarrow B}{\gamma:B}$, where α, β, γ belong to a well-defined language (or, even better, algebra defined over this language) of labels, and where γ would be an appropriate function of α and β. If we were to introduce our original idea of limited-depth inference, then γ could be, e.g., $\max(\alpha, \beta) + 1$ provided that α and β are smaller than some constant N.

A similar idea, although restricted to manipulation of labels which denote

time points, has been introduced in *step-logic* [3] which later evolved into a family of *active logics* [4]. Such a restriction is actually a reasonable first step towards developing a formal system with provable computational properties. Active logics have been used so far to describe a variety of domains, like planning [5], epistemic reasoning [6], reasoning in the context of resource limitations [7] or modeling discourse. We are definitely interested in pursuing this line of investigations, however in a manner that is more amenable to metamathematical investigations. LDS seems to be a perfect technical choice for that purpose. In particular, various possibilities offered by the freedom of choice of the labeling algebras used to define the inference rules can be studied. Properties of the consequence relations defined this way are definitely worth analyzing in order to gather understanding of what can be achieved in the resource-limited setting, and what (semantical) price is paid for this.

3 Related work

The attempts to constrain in a principled way the inference process performed in a logical system have been done as long as one has used logic for knowledge representation and reasoning. One possibility is to limit the expressive power of the first-order logical calculus (as, e.g., it is done in case of description logics) in order to guarantee polynomial-time computability. There is a number of theoretical results in this area (see, e.g., [8]) but we are rather interested in investigations aimed at practical applications and focusing on computational complexity like, e.g., [9, 10, 11].

Another possibility is to use polynomial approximations of the reasoning process. This approach is tightly coupled to the issue of theory compilation. The most important contributions in this area are [1, 12, 13, 14]. However, this approach, although it substantially reduces the computational complexity of the problem, still does not provide tight bounds on the reasoning process.

Yet another possibility is to constrain the inference process in order to retain control over it. An early attempt has been reported in [15]. The next step in this direction was the step-logic [3] that evolved into a family of *active logics* [4]. Such a restriction is actually a reasonable first step towards developing a formal system with provable computational properties. Active logics have been used so far to describe a variety of domains, however, none of the proposed systems has overcome the limitation of the exponential blow-up of the number of formulae produced in the inference process.

One of the main reasons for problems with describing resource-bounded

reasoning is that the formal systems used for this purpose are too powerful. Quite often such a system is based on some propositional or first-order language extended with a modality denoting belief. This immediately leads to the *omniscience problem*: use of any *normal* modal logic equipped with the **K** axiom

$$\models \Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$$

and the necessitation rule

$$\frac{\alpha}{\Box\alpha}$$

will force the agent using it for its reasoning to believe all logical consequences of its current beliefs (see, e.g., [16]). This means that its set of beliefs will necessarily be infinite and it must always be consistent: apparently non-realistic assumptions regarding limited reasoners.

There exists a number of approaches that try to deal with the problem of omniscience. Speaking generally, any such solution must address the need to model bounded resources of agents and, independently, its incomplete reasoning mechanisms. The solutions might be roughly classified into following groups:

- Weakening the system by using a non-standard semantics;
- Formally distinguishing explicit and implicit knowledge;
- Removing closure properties;
- Syntactification.

The first two suffer from partial logical omniscience [17], although they constitute a step in the right direction.

A good example of the second line of thought is the *logic of implicit and explicit belief* proposed by Levesque [15]. His idea consists of distinguishing *explicit* beliefs, i.e., those currently available explicitly in agent's knowledge base; and *implicit* beliefs, i.e., those entailed by explicit beliefs, but not (yet) derived. In order to be able to handle that distinction, the semantics of the classical epistemic logic must be modified. According to Levesque, the usual possible worlds semantics is too coarse-grained, while simple sets of formulae are a too-fine-grained choice. His semantics is based on *situations* which are subsets of possible worlds. The relation between situations is captured by *relevance logic* (where implication is weakened to entailment). Unfortunately, partial omniscience makes this system inappropriate for describing truly limited agents.

The approach of Fagin, Halpern, Moses and Vardi [16], *Interpreted multi-agent systems* captures in a nice way the evolution of an agent's knowledge.

The system consists of the usual knowledge/belief modalities (K_x, indexed by multiple agents involved), mixed with the classical temporal operators (Eventually $\Diamond$, Always $\Box$, Next $\bigcirc$, Until $\mathcal{U}$). The resulting system is interpreted on so called *runs*. The system is still too powerful for our needs, although it may be modified in the direction of non-omniscient agents. The same remarks apply to the recent proposal of van der Hoek and Wooldridge, *Alternating-time Temporal Epistemic Logic (ATEL)*, [18], in which the usual knowledge/belief modalities (K_x) and the classical temporal operators are extended with a dynamic-logic-like concept of *cooperative actions*. The interpretations are based on *concurrent game structures*. Although the authors mention the possibility of describing non-omniscient agents, the main system is developed for at least partially omniscient entities. A similar system, based on active logic, has been proposed by Grant, Kraus and Perlis [19].

The last system we would like to mention in this context is the *Logic of Finite Syntactic Epistemic States* proposed recently by Ågotnes [17]. His system is based on ATEL, but does not assume any structure in the underlying language — the epistemic states of an agent are purely syntactical structures. Knowledge evolution mechanisms are modeled using *rules*; they need not to be necessarily sound nor complete. Although appealing from the formal point of view, this system does not provide any hints about dealing with the computational complexity of the problem.

There is a growing insight that logic, should it be considered as a useful tool for building autonomous intelligent agents, has to be used in a substantially different way than before. Active logics are one example of this insight, while other important contributions might be found, e.g., in [20] or [21].

4 Active Logics

Active logics originated from an attempt to formalize a memory model, inspired by cognitive psychology research, which was studied at the University of Maryland during the 1980s [22]. It has been first formalized by *step logic*. However, this formalisation has left many of the interesting properties of the model outside its scope.

The memory model (MM later on) consists of five parts:

- LTM, the *long term memory*, which contains rules consisting of pairs of formulae: (trigger, contents). 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 recently 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 may provide support for goal-directed behavior.

Three of the parts, LTM, STM and ITM, originate from cognitive psychology research. The other two, QTM and RTM, have been used by Drapkin, Miller and Perlis, as an auxiliary technical device. Figure 1 shows how the parts are connected to each other. Although the details of the model might be discussed, in particular, debating the necessity for modules other than the standard short- and long-term memories, we have based our discussion on the original MM [22], convinced that the auxiliary parts play an important technical role. A more detailed explanation is beyond the scope of this paper and the reader is referred to the original source. It is a matter of future research whether simpler memory models could be formalised in an equally natural way, while retaining the capability of modeling reasoning with resource limitations.

5 Active logics as an LDS

As the first step of testing our ideas we have chosen the first active logic, namely the step logic SL_7 defined in [3]. It is, in its turn, a simplification of MM presented above. It appeared [23] that SL_7 can be rather straightforwardly formulated as an LDS. Below, we show how this formalization can be extended to the original MM. None of the active logic systems defined so far ([7], [24], and [25]) has been able to faithfully capture its full complexity. Therefore our first conclusion is that LDS offers a more expressive mechanism to control deduction. This chapter is based on MM presentation from [22] and $\mathbb{L}_{MM}$ from [26].

5.1 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

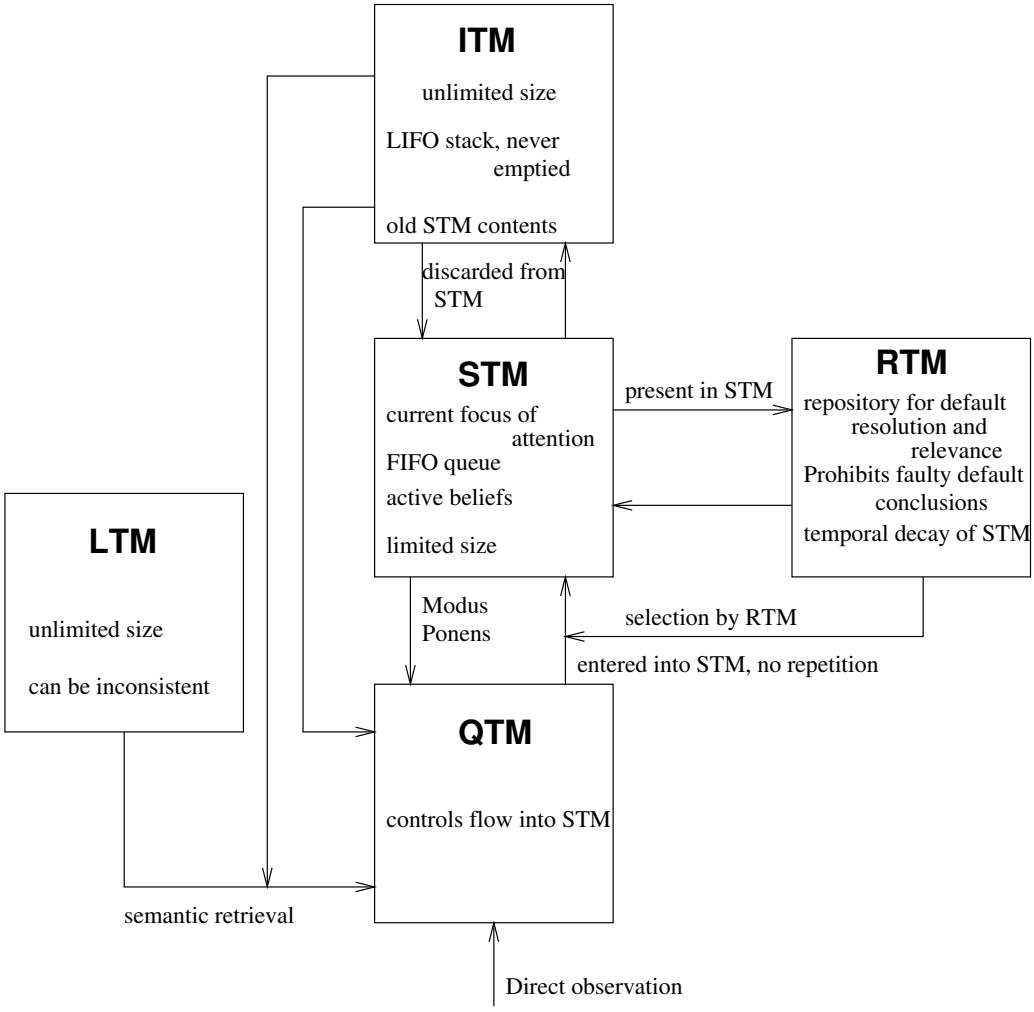

Figure 1: The memory model from [22].

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, *labeled deductive systems*, was presented in [2] as such a unifying framework.

The first step in understanding LDS is to understand the intuitive message, which is the following: Traditional logics manipulate formulae, while an LDS manipulates *declarative units*, i.e., pairs *formula* : *label*. The labels should be viewed as an additional information about the formulae, which is not encoded inside the formulae. E.g., they can contain reliability (in an expert system), where and how a formula was deduced, or time stamps.

A *logic* is here a pair $(\vdash, S_\vdash)$ where $\vdash$ is a structured, possibly non-monotonic consequence relation on a language $\mathbf{L}$ 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, *meta-bases*, in which the labels can be complete 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 and $\mathbb{R}$ is a discipline of labeling formulae of the logic (with labels from the algebra $\mathcal{A}$), together with a notion of a *database* and a family of deduction rules and with agreed ways of propagating the labels via application of the deduction rules.

5.2 Step logics

Step logic is actually not a single system but a family of different 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 of autoepistemic reasoning. The time mechanism allows the on-going process of deduction to be part of the agent's own reasoning. Retractions can be used to implement other forms of non-monotonic reasoning. The logic SL_7 contains all three mechanisms mentioned above.

SL_7 has been rewritten as an LDS in [23]. Below we briefly summarize this construction.

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. Beliefs are parameterized by the time taken for their inference.

Definition 1 (Observation function) *An* observation function *is a function* $OBS : \mathbb{N} \rightarrow \mathcal{P}(S_{wff})$*, 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 2 (History) *A* history *is a finite tuple of belief set/observation set pairs; the sets are finite subsets of* S_{wff}*:* $<< Thm_0, OBS(1) >, < Thm_1, OBS(2) >, ..., < Thm_{i-1}, OBS(i) >>$.

$\mathcal{H}$ will denote 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 3 (Inference function) *An* inference function *is a function* $INF : \mathcal{H} \rightarrow \mathcal{P}(S_{wff})$*, 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 (SL_n-theory) *An* SL_n-theory *over a language* $\mathbf{L}$ *is a triple* $< \mathbf{L}, OBS, INF >$*, where* $\mathbf{L}$ *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* $\mathbf{L}$ *is implicit in the definitions of OBS and INF).*

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

Definition 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.

Definition 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*

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

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

Theorem 1 (Soundness, [3]) *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 $M \models_i \alpha$.*

A problem with step logics is that the set of beliefs may grow rapidly. When the memory model was first formalized in step logic, some of its attractive properties were lost. In particular, in the memory model the short term memory (STM) simulates a focus of attention. The size of the STM limits the number of inferences per step and thus reduces combinatorial explosion which is important in implementations. In step logics this limitation is lost so that the number of formulae in each step may increase exponentially.

5.3 LDS for SL_7

In what follows we present a natural reformulation of SL_7 as an LDS.

As labels we use the natural numbers that represent the time at which a formula has been asserted (observed or deduced), $S_{labels} \stackrel{df}{=} \mathbb{N}$. The *declarative units* of the system are pairs *label* : *formula*. The axioms below express either the time flow or results of observations.

(A1) $i : Now(i)$ for all $i \in \mathbb{N}$ CLOCK

(A2) $i : \alpha$ for all $\alpha \in OBS(i)$, $i \in \mathbb{N}$ OBSERVATIONS

The inference rules used in the system, $\mathbb{R}_{SL_7}$, come from INF_B defined in [3], although in somewhat modified form in order to take account of labels.

(I1) $$\frac{i : \alpha, i : \alpha \rightarrow \beta}{i + 1 : \beta}$$ MP

(I2) $$\frac{i : P_1a, \ldots, i : P_na, i : (\forall x)[(P_1x \wedge \ldots \wedge P_nx) \rightarrow Qx]}{i + 1 : Qa}$$ EMP

I.e., all the prerequisites of the Modus Ponens rules must be present at a time point i in order to assert the conclusion at the time point $i + 1$.

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. $S_{th}(i)$ 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 occurring in them and not occurring as separate theorems (function f_{csf}). The process of finding all closed subformulae for a given finite set of formulae ($f_{formulae}$ yields unlabeled formulae) is computable.

We can now formulate the Negative Introspection rule:

$$\text{(I3)}\quad \frac{\alpha \in f_{csf}(S_{th}(i)), \alpha \notin f_{formulae}(S_{th}(i))}{i+1 : \neg K(i, \ulcorner \alpha \urcorner)} \qquad \begin{array}{c}\textsc{Negative}\\ \textsc{Introspection}\end{array}$$

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.

The (I3) rule involves the knowledge predicate K that takes as one of its arguments a formula. Later rules in this and next sections will introduce predicates *Contra* and *Loses* which behave similarly. In order to keep the language first-order we use the standard reification technique allowing us to treat formulae (or rather their names) as terms of the language. In order to make a distinction between formulae and their names, quoting (shown as $\ulcorner \alpha \urcorner$, for an arbitrary formula α) is used.

Finally, we can define two rules that propagate consistent knowledge onwards:

$$\text{(I4)}\quad \frac{i : \alpha, i : \neg\alpha}{i+1 : \mathit{Contra}(i, \ulcorner \alpha \urcorner, \ulcorner \neg \alpha \urcorner)} \qquad \begin{array}{c}\textsc{Contradiction}\\ \textsc{Detection}\end{array}$$

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

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

Definition 7 $\mathbb{L}_{SL_7} \stackrel{df}{=} (\mathbb{N}, \mathbf{L}, \mathbb{R}_{SL_7})$, *where* $\mathbb{N}$ *denotes the algebra consisting of the set of natural numbers, with the successor (+1) as the only operation,* $\mathbf{L}$ *is the first order language and the consequence relation* $\mathbb{R}_{SL_7}$ *is defined by the inference rules (I1)–(I5)*[1].

[1]Actually, with the restriction that rule (I3) can only be used (meaningfully) when all

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

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

Theorem 2 *A formula α can be derived in the step logic $SL_7(OBS,INF_B)$ at time point i if and only if it can be derived as $\boxed{i:\alpha}$ in the labeled deductive system $\mathbb{L}_{SL_7}$ defined above. 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: see [23].

5.4 Elgot-Drapkin's Memory Model as an LDS

In our opinion the formalisation of MM as step logic is an oversimplification. In particular, the STM size limit is omitted so that the number of formulae in each step may increase rapidly. This problem has also been recognized in [7], [24] and [25], which present other formal active logic systems. However, the major deficiency — the exponential growth of the number of formulae in each reasoning step — has not been satisfactorily solved by any of those approaches. We address this problem again later, postulating a solution.

Below we present an LDS-based formulation of the Memory Model in order to show that LDS has substantially larger expressive power than any of the active logics studied so far.

The labeling algebra is based on the following structure:

$$S_{labels} \stackrel{df}{=} \{LTM, QTM, STM, ITM\} \times S_{wff} \times \{C, U\} \times \mathbb{N}^3 \qquad (1)$$

where the interpretation of a tuple in S_{labels} is the following. If
$(location, trigger, certainity, time, position, time\text{-}left\text{-}in\text{-}rtm) \in S_{labels}$
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 labeled formula should remain in the RTM. $R \in \mathbb{N}$ is a constant used to limit the time a formula remains in RTM after it has left STM.

the possible consequences of (I1), (I2), (I4), (I5) for the time point i have already been drawn. More about this in the next section.

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

(A1')	$(STM, \varepsilon, C, i, i, 0) : Now(i)$	for all $i \in \mathbb{N}$	CLK
(A2')	$(QTM, \varepsilon, C, i, 0, 0) : \alpha$	for all $\alpha \in OBS(i),\ i \in \mathbb{N}$	OBS
(A3')	$(LTM, \gamma, C, 0, 0, 0) : \alpha$	for all rules $(\gamma, \alpha) \in LTM$	LTM

The first rule describes retrieval from LTM into QTM:

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

The relation $\mathcal{R}_{sr}$ describes how the trigger formulae control the semantic retrieval.

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

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

EXTENDED MODUS PONENS

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

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.

The next rule, Negative Introspection, allows one to infer lack of knowledge of a particular formula at time i:

$$\text{(NI)} \quad \frac{\alpha \in f_{csf}(S_{STM}(i)), \alpha \notin f_{formulae}(S_{STM}(i))}{(QTM, \varepsilon, C, i, 0, 0) : \neg K(i, \ulcorner \alpha \urcorner)} \quad \text{NEGATIVE INTROSPECTION}$$

where the set $S_{th}(i)$ described in the previous section is replaced by its memory-bank-specific counterparts, $S_{QTM}(i)$, $S_{new\text{-}STM}(i)$, $S_{STM}(i)$ and $S_{RTM}(i)$. Just like $S_{th}(i)$, they are computable by purely syntactic operations and can be defined recursively on i.

MM in [22] and step logic use different methods to detect and handle contradictions. Step logic indicates detected contradictions with the *Contra* predicate while MM uses instead certainity levels and the *Loses* predicate which is involved in the *RTM* mechanism. We have allowed both possibilities, where CD1 handles the case of equal certainties while CD2 and CD2' deal with the case of different certainties:

$$\text{(CD1)}\quad \frac{\begin{array}{c}(STM,\varepsilon,C,i,p_1,R):\alpha\\(STM,\varepsilon,C,i,p_2,R):\neg\alpha\end{array}}{(QTM,\varepsilon,C,i,0,0):Contra(i,\ulcorner\alpha\urcorner,\ulcorner\neg\alpha\urcorner)}$$

$$\text{(CD2)}\quad \frac{\begin{array}{l}(STM,\varepsilon,c_1,i,p_1,R):\alpha\\(STM,\varepsilon,c_2,i,p_2,R):\neg\alpha\\c_1<c_2\end{array}}{(QTM,\varepsilon,c_1,i,0,0):Loses(\ulcorner\alpha\urcorner)}$$

$$\text{(CD2')}\quad \frac{\begin{array}{l}(STM,\varepsilon,c_1,i,p_1,R):\neg\alpha\\(STM,\varepsilon,c_2,i,p_2,R):\alpha\\c_1<c_2\end{array}}{(QTM,\varepsilon,c_1,i,0,0):Loses(\ulcorner\neg\alpha\urcorner)}$$

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:

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

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 among the QTM contents.

One problem with the lazy STM implementation is that limiting the number of sets in the STM does not necessarily limit the total number of elements in those sets, which is the number of formulae in the STM. If many formulae are moved into STM at the same time step, the sets will contain many elements, the STM will contain many formulae and there will be more computation per inference step. The flow from QTM to STM must thus be controlled to limit the amount of computation to realistic levels. And because there is no selection among the QTM contents, everything that enters QTM also enters STM, so the flow into QTM must be controlled as well.

Our STM implementation uses the position field in the labels. The value of the position field should be zero, unless the associated formula is in STM or ITM. In that case, it contains the time at which the formula was moved into STM by the IQS rule shown below. The position field then remains unchanged, while the IS rule propagates the formula forward in time. A function $f_{min\text{-}STM\text{-}pos}(i)$ computes the minimum position value of all the declarative units in the STM at time i. At time i, the declarative units in STM can have position values $f_{min\text{-}STM\text{-}pos}(i),\ldots,i$, see Figure 2 below.

A simple way to define $f_{min\text{-}STM\text{-}pos}(i)$ would be to set it to $\max(0,i-S+1)$, where S is the intended maximum number of elements in STM. If a position field in a label is $f_{min\text{-}STM\text{-}pos}(i)$ at time i, then the associated formula can be

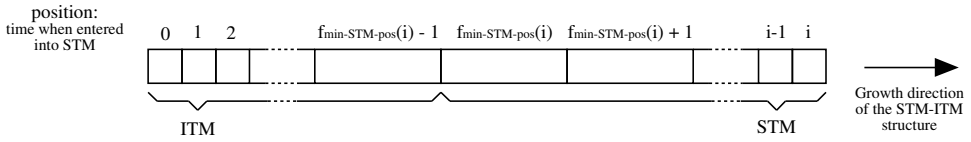

Figure 2: The structure of the STM and ITM buffer.

moved to ITM at time $i+1$. The problem with this simple definition is that formulae will "time out" from STM into ITM, even when no new formulae are entered into STM. That is definitely not the FIFO behaviour described in [22].

Our solution to the "time out" problem is to interpret S as the maximum number of *non-empty* sets in STM. We use a more complex definition of $f_{min\text{-}STM\text{-}pos}(i)$ and do not move anything out from STM to ITM if nothing is moved in from QTM to STM. The exact $f_{min\text{-}STM\text{-}pos}(i)$ definition, rather cumbersome, is omitted but can be found in [26].

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:

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

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 $f_{min\text{-}STM\text{-}pos}$ mentioned above:

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

INHERITANCE IN STM

$$\text{(IS)}\quad \frac{\begin{array}{l}(STM, \varepsilon, c, i, p, R) : \alpha \\ (p > f_{min\text{-}STM\text{-}pos}(i)) \vee (S_{new\text{-}STM}(i+1) = \emptyset) \\ Contra(i-1, \ulcorner\alpha\urcorner, \ulcorner\beta\urcorner) \notin f_{formulae}(S_{STM}(i)) \\ Contra(i-1, \ulcorner\beta\urcorner, \ulcorner\alpha\urcorner) \notin f_{formulae}(S_{STM}(i)) \\ Loses(\ulcorner\alpha\urcorner) \notin f_{formulae}(S_{STM}(i)) \\ (\alpha \neq Now(i)) \wedge (\alpha \neq K(i-1, \ulcorner\beta\urcorner)) \vee (K(i, \ulcorner\beta\urcorner) \notin S_{QTM}(i)) \\ (\alpha \neq Contra(i-1, \ulcorner\beta\urcorner, \ulcorner\gamma\urcorner)) \vee (Contra(i, \ulcorner\beta\urcorner, \ulcorner\gamma\urcorner) \notin S_{QTM}(i))\end{array}}{(STM, \varepsilon, c, i+1, p, R) : \alpha}$$

INHERITANCE STM $\rightarrow$ ITM

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

It should be pointed out that the sets of formulae contained in the *STM* can grow at most polynomially, avoiding the combinatorial explosion of active logics. On the other hand, the size of sets in each position of the *STM* may still be quite large.

We can now define the LDS encoding the memory model:

Definition 8 (Memory model LDS) $\mathbb{L}_{MM} \stackrel{df}{=} (S_{labels}, \mathbf{L}, \mathbb{R}_{MM})$*, where the consequence relation* $\mathbb{R}_{MM}$ *is defined by the rules (SR), (MP), (EMP), (NI), (CD1), (CD2), (IL), (IQS), (IS), (ISI) and (II).*

S_{labels} should be interpreted as an algebra.

The next step would be to show that the behaviour of $\mathbb{L}_{MM}$ is indeed as expected, namely faithfully implements the behaviour of MM. Unfortunately, it cannot be done in a formal way because the original memory model [22] was introduced only as an informal description of a cognitively plausible reasoning mechanism. Although this model, according to the authors, has been tested in practice, it has never been completely formalised. The subsequent formal systems, step logic and a number of active logics based on it, do not have all the control mechanisms present in the original MM. Therefore the correspondence could only be established against our interpretation of the behaviour as described in the literature. In order to achieve such result we have interpreted the $\mathbb{L}_{MM}$ system using structures resembling the ones illustrated in Figure 1.

One problem with $\mathbb{L}_{MM}$ 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 only sometimes agree with 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.

The sets 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 part of the current database. But when describing the system as an algebraic LDS we can't be sure of the "completeness level" of an arbitrary database. The requirement for completeness requires restrictions on the order in which inference rules are applied; some of the rules can't be applied to some of the declarative units until the database has reached a certain level of completeness.

One of the strengths of LDS is that it can handle features which are normally at the meta-level, at the object level. It turns out that it can handle this ordering of the application of inference rules, too. The trick consists of including the whole database in the labeling of formulae. The details of such solution are presented in [26].

6 Planning in real-time

Some work on active logics has been devoted to application of this approach in real-time planning. In particular, Nirkhe [24, 7] has introduced an active logic for reasoning with limited resources and provided a modal semantics for it. However, the computational issues have not been addressed by this research.

A later formulation in [5] describes an active logic-based system, called Alma/Carne, designed for planning and plan execution in real-time. Alma is the reasoning part of the agent, based on active logic, and capable to deal with deadlines. Carne is the plan execution module, procedural in its nature, cooperating with Alma. However, the computational complexity problem, inherited from step logic, has not been addressed here either.

Although the idea behind Alma/Carne is appealing — in some sense it is rather obvious that a decent real-time cognitive system should have such a structure — the limitations of active logic, consisting of low granularity, limited to time points, of the reasoning process, are putting the possibility of practical applications of this approach into question.

As we have shown above, $\mathbb{L}_{MM}$ can offer exactly the same functionality as active logics, but with much richer structure of the labels attached to formulae. This way we can limit the number of formulae staying in focus to a small, manageable value. We can introduce a labeling in which not only time points are relevant for predicting the real-time behaviour of the system, but where the individual applications of inference rules can be counted, timed and predicted, if necessary. Therefore a solution similar to Alma/Carne, but based on $\mathbb{L}_{MM}$ (or some other suitable LDS) as the reasoning formalism, is envisioned as a possible breakthrough leading to the hard-real-time predictability of a reasoning system. The next step would be to perform the worst-case execution time analysis of the reasoning process, similarly to the one proposed in [27] for a different system.

As the first step in this direction we are developing a prototype implementation of a theorem prover for LDS-based systems, where the labeling policy and the "classical part" of an inference rule are handled in a modular way, allowing one to exchange the label processing policy while retaining intact those parts of inference rules (e.g., Modus Ponens or Inheritance) that deal with the actual formulae. The system will provide a framework for experimenting with different LDS-s, analyzing their computational properties, and leading to a formalization that can survive the requirements of real-time. The prototype has been so far applied to simple problems, solvable in principle by hand. But already at this stage we see the benefit of the prover as a proof verifier. In particular, we have redone in $\mathbb{L}_{MM}$ the proofs from [22], illustrating the defeasible reasoning patterns. These proofs have been then verified by the prototype implementation of the automatic LDS prover, actually finding a couple of errors unnoticed earlier. As each of the proofs is approximately 100 steps long, we have decided not to include them here. The interested reader may find them in [26]

7 Conclusions and future work

We have presented an LDS-based formalization for the memory model entailing later formal active logic systems. This allows us to expect that even in the case of more complex, time-limited reasoning patterns, LDS will appear to be a useful and powerful tool. Actually, the technical problem with restricting the inference rule applications to a particular order in order to get hold of non-monotonic dependencies, can be solved satisfactorily by just extending the labeling algebra and then constraining the inference rule invocations by appropriately constructed predicates over these labels. LDS provides also far more sophisticated basis for defining semantics of such resource-limited

reasoners, in particular, systems that reason in time and about time.

The technique described in this paper raises a number of interesting questions that we intend to investigate. First, what is the actual status of the consequence relation $\mathbb{R}_{MM}$ in the spectrum of algebraic consequence relations defined in [2]? Can this knowledge be used to better characterize the logic it captures? Is it possible to characterize the time-limited reasoning in such manner that the worst-case reasoning time (analogously to the worst-case execution time, known from the area of real-time systems) could be effectively computed? What would be then the semantical characterization of such a logic?

Another challenging problem is to practically realize a planning system based on this approach. We expect to be able to implement a $\mathbb{L}_{MM}$-based planner in the near future, and to experiment with physical robots in the next stage of the project.

Speaking slightly more generally, we hope that LDS may serve as a tool for specifying logics that artificial intelligence is looking for: formalisms describing the *knowledge in flux* (to quote the famous title of Peter Gärdenfors) that serve intelligent agents to reason about the world they are embedded in and about other agents, in real-time, without resorting to artificial, extralogical mechanisms.

Acknowledgments

The authors are grateful to the anonymous reviewers for their detailed comments that led to improvements of this article.

The second author would like to thank Michael Fisher for pointing out LDS mechanism as a potential tool for implementing time-limited reasoning.

Sonia Fabre Escusa has made the preliminary implementation of a theorem prover for the $\mathbb{L}_{MM}$ LDS. It allowed us to find a number of inaccuracies in the original text.

References

[1] B. Selman and H. Kautz. Knowledge compilation and theory approximation. *JACM*, 43(2):193–224, 1996.

[2] D. Gabbay. *Labelled Deductive Systems, Vol. 1.* Oxford University Press, 1996.

[3] J. Elgot-Drapkin. *Step Logic: Reasoning Situated in Time*. PhD thesis, Department of Computer Science, University of Maryland, 1988.

[4] J. Elgot-Drapkin, S. Kraus, M. Miller, M. Nirkhe, and D. Perlis. Active logics: A unified formal approach to episodic reasoning. Technical report, Department of Computer Science, University of Maryland, 1996.

[5] K. Purang, D. Purushothaman, D. Traum, C. Andersen, D. Traum, and D. Perlis. Practical reasoning and plan execution with active logic. In *Proceedings of the IJCAI'99 Workshop on Practical Reasoning and Rationality*, 1999.

[6] J. Elgot-Drapkin. Step-logic and the three-wise-men problem. In *Proc. AAAI*, pages 412–417, 1991.

[7] M. Nirkhe, S. Kraus, and D. Perlis. Situated reasoning within tight deadlines and realistic space and computation bounds. In *Proc. Common Sense 93*, 1993.

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

[9] G. De Giacomo, L. Iochhi, D. Nardi, and R. Rosati. A theory and implementation of cognitive mobile robots. *J. Logic Computation*, 9(5):759–785, 1999.

[10] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. In *Proc. IJCAI 85*, pages 455–458, 1985.

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

[12] M. Cadoli and F. Donini. A survey on knowledge compilation. *AI Communications*, 2001.

[13] M. Cadoli and M. Schaerf. Approximate reasoning and non-omniscient agents. In *Proc. TARK 92*, pages 169–183, 1992.

[14] G. Gogic, C. Papadimitriou, and M. Sideri. Incremental recompilation of knowledge. *JAIR*, 8:23–37, 1998.

[15] H. Levesque. A logic of implicit and explicit belief. In *Proc. AAAI 84*, pages 198–202, 1984.

[16] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 2003.

[17] T. Ågotnes. *A Logic of Finite Syntactic Epistemic States*. PhD thesis, Department of Informatics, University of Bergen, Norway, 2004.

[18] W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75:125–157, 2003.

[19] J. Grant, S. Kraus, and D. Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, 3(4):455–458, 2000.

[20] D. Gabbay and J. Woods. The new logic. *L. J. of the IGPL*, 9(2):141–174, 2001.

[21] M. Wooldridge and A. Lomuscio. A computationally grounded logic of visibility, perception, and knowledge. *L. J. of the IGPL*, 9(2):257–272, 2001.

[22] J. Drapkin, M. Miller, and D. Perlis. A memory model for real-time commonsense reasoning. Technical Report TR-86-21, Department of Computer Science, University of Maryland, 1986.

[23] M. Asker and J. Malec. Reasoning with limited resources: An LDS-based approach. In et al. B. Tessem, editor, *Proc. Eight Scandinavian Conference on Artificial Intelligence*, pages 13–24. IOS Press, 2003.

[24] M. Nirkhe. *Time-Situated Reasoning Within Tight Deadlines and Realistic Space and Computation Bounds*. PhD thesis, Department of Computer Science, University of Maryland, 1994.

[25] A. Globerman. A modal active logic with focus of attention for reasoning in time. Master's thesis, Department of Mathematics and Computer Science, Bar-Illan University, 1997.

[26] M. Asker. Logical reasoning with temporal constraints. Master's thesis, Department of Computer Science, Lund University, August 2003. Available at `http://ai.cs.lth.se/xj/MikaelAsker/exjobb0820.ps`.

[27] M. Lin and J. Malec. Timing analysis of RL programs. *Control Engineering Practice*, 6:403–408, 1998.