On Reasoning and Planning in Real-Time: An LDS-Based Approach

Mikael Asker and **Jacek Malec**
Department of Computer Science
Lund University
Box 118
221 00 LUND, Sweden
email: `mikael.asker@fagotten.org, jacek@cs.lth.se`

Abstract

Reasoning with limited computational resources (such as time or memory) is an important problem, in particular in cognitive 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 still do not offer the ultimate 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 shown that the LDS-based approach is strictly more expressive than active logics. We have also implemented a prototype automatic theorem prover for LDS-based systems.

Introduction

Reasoning with limited computational resources (such as time or memory) is an important problem, in particular in cognitive embedded systems. Usually a decision, based on a principled reasoning process, needs to be taken within limited time and given constraints on 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: classical 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 occured in Artificial Intelligence in the middle of 1980s can be attributed to increasing awareness of those limitations of the the 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.

One of the more interesting lines of research in this area during 1990s has focused on logic as a model of an ongoing reasoning process rather than as a static characterization of contents of a knowledge base. It begun with step-logic (Elgot-Drapkin 1988) and evolved into a family of *active logics*. The most recent focus of this research is on modeling dialog and discourse. However, other interesting applications like planning or multi-agent systems have also been investigated, while some other possibilities wait for analysis. In particular, the possibility of applying this framework to resource-bounded reasoning in cognitive robotic 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. After presenting the background of our investigation we introduce the memory model being the foundation for active logics research. The following section presents an LDS formalization of the memory model. Then we discuss how the described approach could be used for planning in real-time for robotic applications. After that we briefly present related work. Finally the conclusions and some suggestion of further work are presented.

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 (Selman & Kautz 1996). 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 reappear in various disguises in our investigations.

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 several previous approaches 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 (Gabbay 1996).

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* (Elgot-Drapkin 1988) which later evolved into a family of *active logics* (Elgot-Drapkin *et al.* 1996). 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 (Purang *et al.* 1999), epistemic reasoning (Elgot-Drapkin 1991), reasoning in the context of resource limitations (Nirkhe, Kraus, & Perlis 1993) 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.

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 (Drapkin, Miller, & Perlis 1986). It has been first formalized by *step logic*. However, this formalization 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.

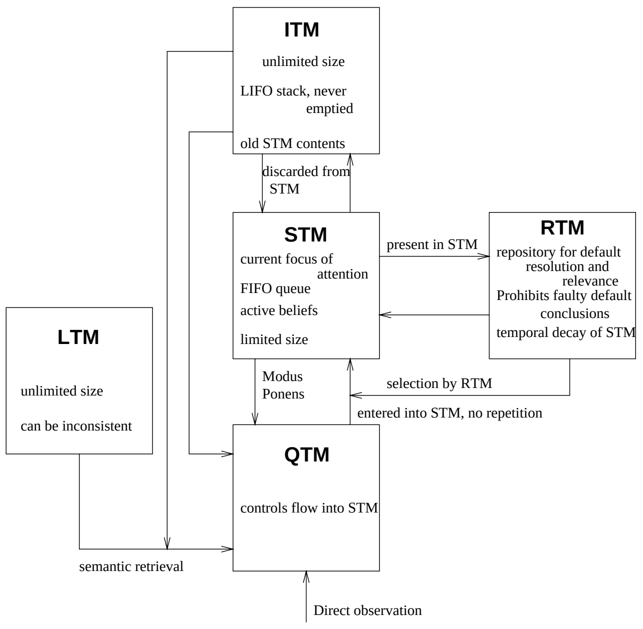

Figure 1: The memory model from (Drapkin, Miller, & Perlis 1986).

- 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 invented by Drapkin, Miller and Perlis, as an auxiliary technical device. Figure 1 shows how the parts are connected to each other.

Active logics as LDSs

As the first step we have chosen the first active logic, namely the step logic SL_7 defined in (Elgot-Drapkin 1988). It is, in its turn, a simplification of MM presented above. It appeared (Asker & Malec 2003) 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 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 section is based on MM presentation from (Drapkin, Miller, & Perlis 1986) and $\mathbb{L}_{MM}$ from (Asker 2003).

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, *labeled deductive systems*, was presented in (Gabbay 1996) 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, while an LDS manipulates *declarative units*, i.e., pairs $formula : label$, of formulae and labels. The labels should be viewed as more 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 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 and $\mathbb{R}$ is a discipline of labeling 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 application of the deduction rules.

Elgot-Drapkin's Memory Model as an LDS

In our opinion the formalization of MM in 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 (Nirkhe, Kraus, & Perlis 1993) and (Globerman 1997), 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. In what follows we address this problem again, 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:

$$S_{labels} \overset{df}{=} \{LTM, QTM, STM, ITM\} \times S_{wff} \times \{C, U\} \times \mathbb{N}^3$$

where the interpretation of a tuple in S_{labels} is the following. If $(loc, trig, cert, time, pos, \textit{time-rtm}) \in S_{labels}$ is a label, then *loc* encodes the memory bank location of the formula (one of LTM, QTM, STM or ITM), *trig* is used for encoding the triggering formula for LTM items (in particular, ε is used to denote the empty triggering formula), *cert* is used in case of defeasible reasoning to encode the status of the formula (certain or uncertain), *time* is the inference time, *pos* denotes the formula's position in STM or ITM, and, finally, *time-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 set of axioms, S_{axioms}, is determined by the following three schemata:

$(STM, \varepsilon, C, i, i, 0) : Now(i)$	for all $i \in \mathbb{N}$ (CLOCK)
$(QTM, \varepsilon, C, i, 0, 0) : \alpha$	for all $\alpha \in OBS(i)$, $i \in \mathbb{N}$ (OBS)
$(LTM, \gamma, C, 0, 0, 0) : \alpha$	for all $(\gamma, \alpha) \in LTM$ (LTM)

The first rule, SEMANTIC RETRIEVAL (SR), describes retrieval from LTM into QTM:

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

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

The "real" inference using either MODUS PONENS (MP) or EXTENDED MODUS PONENS (EMP) is performed from STM to QTM:

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

$$\frac{\begin{array}{l}(STM, \varepsilon, c_1, i, p_1, R) : P_1 a \\ \dots \\ (STM, \varepsilon, c_n, i, p_n, R) : P_n a \\ (STM, \varepsilon, c_0, i, p_0, R) : (\forall x)[(P_1 x \wedge \ldots \wedge P_n x) \to Qx]\end{array}}{(QTM, \varepsilon, min(c_0, c_1, \ldots, c_n), i, 0, 0) : Qa}$$

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 (NI), 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 (CLOCK), 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 occuring 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 NI rule:

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

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

The NI rule involves the knowledge predicate K that takes as one of its arguments a formula. Later rules 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.

MM in (Drapkin, Miller, & Perlis 1986) and step logic use different methods to detect and handle contradictions. Step logic indicates detected contradictions with the $Contra$ predicate while MM uses instead certainty 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 deals with the case of different certainties:

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

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

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:

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

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. The position field then remains unchanged, while the IS rule propagates the formula forwards in time. A function $f_{\textit{min-STM-pos}}(i)$ 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 $f_{\textit{min-STM-pos}}(i), \ldots, i$, see Figure 2.

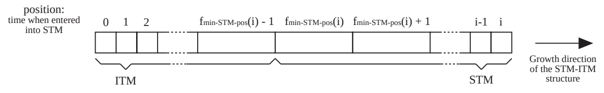

Figure 2: The structure of the STM and ITM buffer.

A simple way to define $f_{\textit{min-STM-pos}}(i)$ 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 $f_{\textit{min-STM-pos}}(i)$ at time i, then the associated formula can be 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.

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 $f_{\textit{min-STM-pos}}(i)$ and do not move anything out from STM to ITM if nothing is moved in from QTM to STM. The exact definition, rather cumbersome, is omitted here but can be found in (Asker 2003).

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:

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

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

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

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

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

We can now define the LDS encoding the memory model:

Definition 1 (MM LDS) $\mathbb{L}_{MM} \overset{df}{=} (S_{labels}, \mathbf{L}, \mathbb{R}_{MM})$, *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 (Drapkin, Miller, & Perlis 1986) 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 one can interpret the $\mathbb{L}_{MM}$ system using structures resembling the ones illustrated in Figure 1. This is the approach we have taken in our investigations.

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 requirement for completeness restricts 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. A sketch of such solution is presented in (Asker 2003).

Planning in real-time

Some work on active logics has been devoted to application of this approach in real-time planning. In particular, Nirkhe (Nirkhe, Kraus, & Perlis 1993) 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 (Purang *et al.* 1999) 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 issue inherited from the 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.

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 those parts of inference rules (e.g., Modus Ponens or Inheritance) that deal with the actual formulae, intact. The system will provide a framework for experimenting with different LDSs, 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.

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., in description logics) in order to guarantee polynomial-time computability. There is a number of theoretical results in this area (see, e.g., (Ebbinghaus 1999)) but we are rather interested in investigations aimed at practical computational complexity. One of the more popular approaches is to use a restricted language (like, e.g., description logics), see (Giacomo *et al.* 1999; Patel-Schneider 1985; 1986) for examples of this approach in practice.

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 (Selman & Kautz 1996; Cadoli & Donini 2001; Cadoli & Schaerf 1992; Gogic, Papadimitriou, & Sideri 1998). 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 (Levesque 1984). The next step in this direction was the step-logic (Elgot-Drapkin 1988) that evolved into a family of *active logics* (Elgot-Drapkin *et al.* 1996). Such a restriction is actually a reasonable first step towards developing a formal system with provable computational properties. However, none of the AL systems has overcome the limitation of the exponential blow-up of the number of formulae produced in the inference process.

There is a growing insight that logic, if it is to 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 (Gabbay & Woods 2001) or (Wooldridge & Lomuscio 2001).

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 (Gabbay 1996)? 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, extra-logical mechanisms.

Acknowledgments

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 text.

References

Asker, M., and Malec, J. 2003. Reasoning with limited resources: An LDS-based approach. In B. Tessem, e. a., ed., *Proc. Eight Scandinavian Conference on Artificial Intelligence*, 13–24. IOS Press.

Asker, M. 2003. Logical reasoning with temporal constraints. Master's thesis, Department of Computer Science, Lund University. Available at http://ai.cs.lth.se/.

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

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

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

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

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

Elgot-Drapkin, J. 1988. *Step Logic: Reasoning Situated in Time*. Ph.D. Dissertation, Department of Computer Science, University of Maryland.

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

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

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

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

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

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

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

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

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

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

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

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

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