Active Logic, Labeled Deductive Systems, reasoning with limited resources

Mikael Asker, Jacek Malec*

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

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 inapropriate 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 reformulated a pair of systems, namely the *memory model* and its simplification, the *step logic*, proposed by Elgot-Drapkin, as Labeled Deductive Systems. This paper presents our motivation behind this project and presents in some detail the first system.

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 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: 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. More details about each of those lines of research can be found in Section 5.

One of the more interesting lines of 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

*Department of Computer Science, Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden, `mikael.asker@fagotten.org, jacek@cs.lth.se`

of contents of a knowledge base. It begun with step-logic [7] 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 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. Section 3 introduces the memory model being the foundation for active logics research. Then Section 4 presents an LDS formalization of the memory model. In Section 5 we briefly discuss related work. 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 [21]. 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 it far from obvious. We have investigated previous approaches (listed in Section 5) 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 [10].

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 $\frac{A, A \to B}{B}$ a labeled deduction system would use $\frac{\alpha : A,\ \beta : A \to B}{\gamma : B}$, 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* [7] which later evolved into a family of *active logics* [9]. 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 [20], epistemic reasoning [8], reasoning in the context of resource limitations [17] 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 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 [5]. 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 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, originate from cognitive pychology 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.

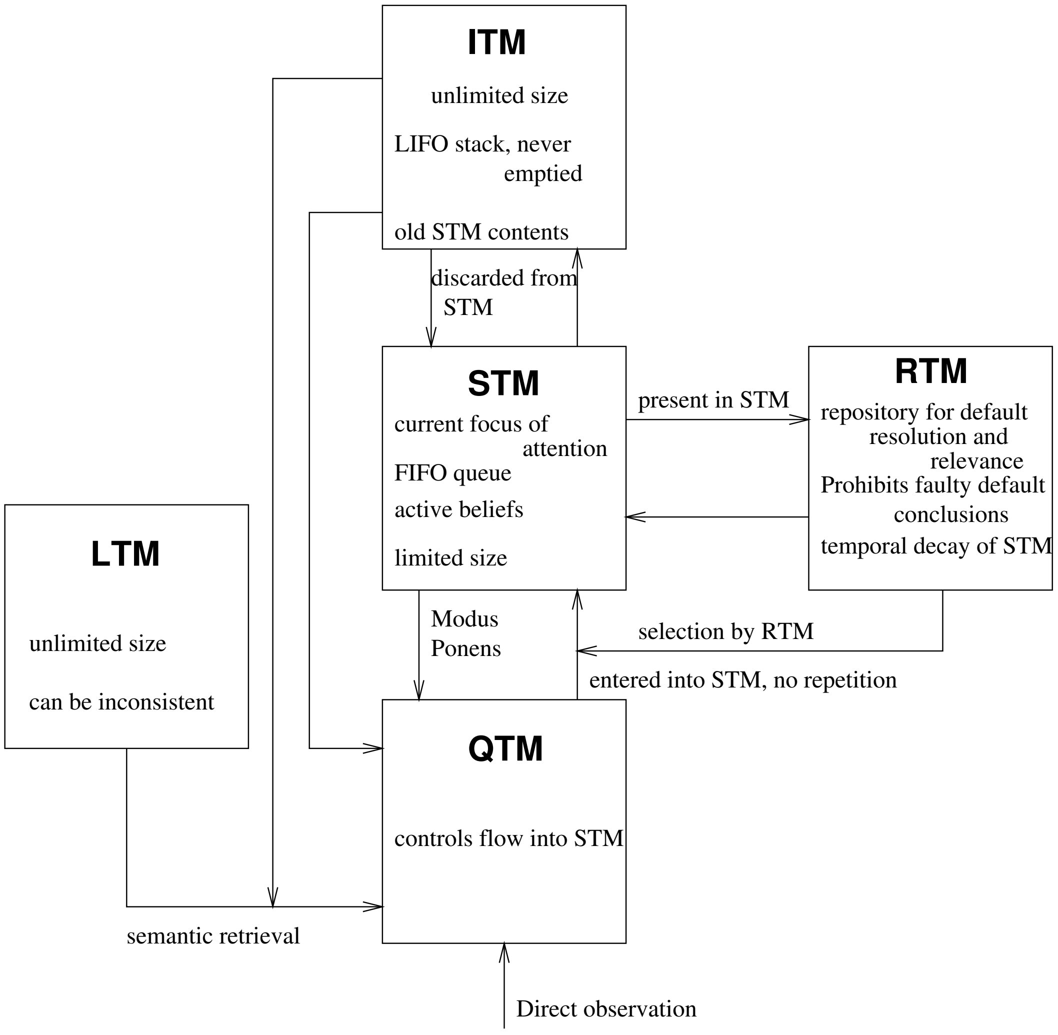

Figure 1. The memory model from [5].

4 Active logics as an LDS

As the first step we have chosen the first active logic, namely the step logic SL_7 defined in [7]. It is, in its turn, a simplification of MM presented above. It appeared [2] that SL_7 can be rather straightforwardly formulated as an LDS. Below, we show how this formalization can be extended to the original MM. Due to the space limitation we are able to present only rudimentary introduction to the original systems. The reader is asked to refer to the original publications for detailed presentation of each of them: MM [5] and LDS [10]. a more detailed presentation can be found in [1].

4.1 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 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 [10] 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*, 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.

4.2 Elgot-Drapkin's memory model as an LDS

In our opinion the formalisation 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 [17], [16] and [13]. Therefore we have attempted to faithfully model the original system.

The labeling algebra is based on the following structure:

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

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), *certainity* 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. $R \in \mathbb{N}$ 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:

(A1')	$(STM, \varepsilon, C, i, i, 0) : Now(i)$	for all $i \in \mathbb{N}$	CLOCK
(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:

$$\text{(MP)} \quad \frac{(STM, \varepsilon, c_1, i, p_1, R) : \alpha, (STM, \varepsilon, c_2, i, p_2, R) : \alpha \to \beta}{(QTM, \varepsilon, min(c_1, c_2), i, 0, 0) : \beta} \quad \textsc{Modus ponens}$$

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

where function *min* is defined over the set $\{U, C\}$ of certainity 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. 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 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 Negative Introspection rule:

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

where the set $S_{th}(i)$ described above 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 [5] 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[1]:

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

[1]Actually we need also a rule symmetric to (CD2).

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

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. To achieve this behaviour we introduce a function $f_{\mathit{min\text{-}STM\text{-}pos}}(i)$ which counts the number of positions the *STM* structure occupies in the queue containing *STM*, *QTM* and *ITM*in the same time. The exact definition, rather cumbersome, is omitted but can be found in [1].

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, meaning more computation per inference step. The flow from QTM to STM must thus 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:

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

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

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

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

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

We can now define the LDS encoding the memory model:

Definition 1 (Memory model 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. As this requires additional clarifications, we do not present the final theorem, expecting the reader to have the intuition of its contents. As usual, the details can be found in [1].

One problem with $\mathbb{L}_{MM}$ is that the functions $S_{th}(i)$, $S_{QTM}(i)$, $S_{STM}(i)$, $S_{\textit{new-STM}}(i)$ 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 metalevel, 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. Details of such solution are presented in [1].

5 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., [6]) 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 [12, 18, 19] 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 [21, 3, 4, 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 [7] that evolved into a family of *active logics* [9]. 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 [20], epistemic reasoning [8], reasoning in the context of resource limitations [17] or modeling discourse.

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 [11] or [22].

6 Conclusions and future work

We have presented an LDS-based formalization for an early active logic, based on the memory model. 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 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.

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 [10]? 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?

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, without resorting to artificial, extra-logical mechanisms.

References

[1] 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`.

[2] 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.

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

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

[5] 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.

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

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

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

[9] 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.

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

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

[12] 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.

[13] 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.

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

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

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

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

[20] 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.

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

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