Electronic News Journal on Reasoning about Actions and Change

Editor: Erik Sandewall

Volume 2, Number 6

Contributions during June, 1998

ISSN, print: 1403 - 2031
on-line: 1403 - 204X

Linköping University Electronic Press
http://www.ep.liu.se/ej/enrac/1998/06/

Publication Trail and Update Rules

The Electronic News Journals (ENJ) are a medium for exchange of scientific information and debate. In particular, they serve as the forum where articles received by the ETAI (Electronic Transactions on Artificial Intelligence) are discussed publicly for review, in separate News Journals for each ETAI area.

The Electronic News Journal on Reasoning about Actions and Change (ENRAC), in particular, uses a publication trail where contributed information is first distributed on a daily basis as *Newsletters* using electronic mail and web pages in HTML. Then, contributions during the same month are compiled into an HTML version of the monthly *News Journal*. Finally, the same information is converted via Latex to a postscript version that is suitable for printing on paper and reading off-line. It is formally published and archived by the Linköping University Electronic Press in both its electronic and its paper form, as described on page (iv).

Offprints of individual segments. Each News Journal issue consists of a sequence of *segments* relating to specific topics. "Offprints" of these can be obtained from the URL:s indicated at the head of the first page of each segment, or from the ENRAC service page specified below. It is intended to maintain these URL:s and their contents for the foreseeable future. Misprints of minor significance are corrected in these offprints (but not in the E-Press version since it is considered archival).

Corresponding HTML edition with dynamic links. The HTML edition contains considerable numbers of links to other pages and structures on the net: articles that are available on-line, home pages of conferences and of individual researchers, links to other part of the ETAI structure, and so on. Due to the natural limitations of the paper medium, only some of the WWW links have been retained here as footnotes. In order to retrieve articles and other information that are referenced in the present issue, it is recommended to look up the corresponding issue in HTML and to use its link. The HTML issues of the News Journals on Reasoning about Actions and Change can be found at the ENRAC *service page,*

`http://www.etaij.org/rac/nj/`

It is intended that the HTML issues will by updated continously to the largest extent possible, for example by replacing URL links to the home pages of authors that have changed to another site.

Table of Contents

Dates of Publication

Since the date of publication may be understood either as the date of first public appearance, or as the day of reproduction on paper in many copies, and since both of these definitions may be difficult to apply in the case of electronic publication, we make the following clarifying statement.

The contents of the present issue were put on-line in their original, HTML version during the month of June, 1998. Then the contents were edited and formatted, resulting in the present, formatted version which was published on 16.7.1999, in two concurrent editions: an on-line edition and a paper edition. The on-line edition was timestamped electronically and put on-line by Linköping University Electronic Press at the URL specified on page (i). The paper edition was obtained by printing the on-line edition on a standard computer printer. It was reproduced in 25 copies, legally archived, and made available for distribution.

Electronic News Journal on Reasoning about Actions and Change 213
Vol. 2: *213-213*, 30.6.1998
http://www.etaij.org/rac/nj/9806/

The Month of June

Summary of News Journals Contents

Three additional articles were submitted during June to the present ETAI area, and discussion about them started. The topical debate on *ontologies of time* that raged during the earlier months, took a timeout in June in order to resolve some possible misunderstandings off-line.

The present ENRAC issue contains a small number of pages compared to the previous ones. This is perfectly in order. In fact, one of the advantages of an electronic and freely available periodical is that we can allow the size of issues to vary according to the need; we feel no obligation to "fill" the same size of issue every time.

Vol. 2: *214-218*, 30.6.1998
http://www.etaij.org/ra/rac/007/

Protocol of on-line discussion during June, 1998
about the following research article:

Iliano Cervesato, Massimo Franceschet, and Angelo Montanari

The Complexity of Model Checking in Modal Event Calculi with Quantifiers

Q2. **Peter Jonsson** (27.5)

Comments and questions to the authors:

Page 4, col 1, para 2: Please find a better name than "condition ()".

Page 6, col 1, para 3: You write "The best approximation of /formula/ we can achieve...". In what sense is it the "best" approximation?

Chapter 4 "Implementation". I agree that the implementation of QCMEC in λ-prolog is very elegant. However, due to my own experience of prolog-like languages, I wonder how efficient the implementation is? I suggest that you include a small empirical investigations as follows: First, implement a QCMEC algorithm in some fast, imperative language such as C. This is straightforward since you have already proved that QCMEC is in PSPACE and your way of showing this (by exhibiting a polynomial-time alternating TM) immediately suggests a recursive algorithm for the QCMEC problem. (Even though such a naive algorithm may not be the fastest algorithm, it is probably efficient enough.) Then, compare your implementation in λ-prolog with the C-implementation on a number of test examples (such as the diagnosis example). Even though this experiment does not prove anything, it can very well give us an idea of the (in)efficiency of the implementations. I will not be surprised if the C-implementation completely outperforms the λ-prolog-implementation. I will be extremely surprised if the λ-prolog-implementation outperforms the C-implementation.

Page 8, before Th. 4.1: You write "We only show the statement of our soundness and completeness result since a fully worked out proof would require a very detailed account of the semantics of λ-prolog". So why implement QCMEC in λ-prolog at all? By implementing a straightforward QCMEC algorithm using a language with a more intuitive semantics, the soundness/completeness proof would be more or less trivial (once again, compare with your PSPACE-membership proof of QCMEC).

Chapter 5 "Complexity Analysis" Why not rearrange the section as follows: Begin the section by proving that QCMEC is in PSPACE. Then there is no need for first showing that CMEC is in PSPACE, extending the result to EQCMEC and finally realize that the proof for EQCMEC also works for QCMEC.

Page 8, col 2, proof of Th. 5.3.: You write "...the algorithm enters in an AND (resp. OR) state. It nondeterministically choose one among α and β and evaluates it in w." This should be rephrased. If you are in an OR state, the sentence is correct but if you are in an AND state, no nondeterministical choice takes place. Instead, the ATM checks that all paths starting in the AND state is an accepting path.

Page 10, col 1, end of proof of Th. 5.3. Please give the proof of why $w \vdash F_1$ iff G is true. Or, at least, give the reader a hint...

Page 11, col 1: You write "In Section 4 we have transliterated the definition of QCMEC and its subcalculi in the higher-order logic programming λ-prolog. The directness of the implementation allows checking easily that the complexity of the implemented algorithms coincide with the bounds proved in this section for the problems they implement". One should be extremely careful when justifying complexity results with implementations in logic programming languages. Such languages often introduce large overheads and it is difficult to know the exact complexity of the different operations (unification,...) that are performed by the systems. Your assertion may very well be correct, but it needs further justification.

A2. **Angelo Montanari** (14.6)

> *Page 4, col 1, para 2: Please find a better name than "condition (*)".*

Ok. Let us call it "singleton condition" since both the initiate and terminate maps associate each property with a singleton set (or an empty set).

> *Page 6, col 1, para 3: You write "The best approximation of /formula/ we can achieve...". In what sense is it the "best" approximation?*

Ok. Since we are not interested in providing a formal measure of the goodness of an approximation, we will rewrite the sentence as follows: "Let us consider the following approximation of $\Diamond\varphi^i$: ... which in general is not equivalent to, but only implied by, $\Diamond\varphi^i$."

> *Chapter 4 "Implementation". I agree that the implementation of QCMEC in λ-prolog is very elegant. However, due to my own experience of prolog-like languages, I wonder how efficient the implementation is? I suggest that you include a small empirical investigations as follows: First, implement a QCMEC algorithm in some fast, imperative language such as C. This is straightforward since you have already proved that QCMEC is in PSPACE and your way of showing this (by exhibiting a polynomial-time alternating TM) immediately suggests a recursive algorithm for the QCMEC problem. (Even though such a naive algorithm may not be the fastest algorithm, it is probably efficient enough.) Then, compare your implementation in λ-prolog with the C-implementation on a number of test examples (such as the diagnosis example). Even though this experiment does not prove anything, it can very well give us an idea of the (in)efficiency of the implementations. I will not be surprised if the C-implementation completely outperforms the λ-prolog-implementation. I will be extremely surprised if the λ-prolog-implementation outperforms the C-implementation.*

We presented a lambda-Prolog implementation of QCMEC for two reasons: first to show that this formalism is implementable, and quite easily so; second to promote in the AI community languages and a style of programming that permit direct encodings of the problem at hand, to the point that it is feasible to provide rigorous mathematical proofs of correctness (see below).

We did not consider efficiency issues. We would ourselves be very surprised if our lambda-Prolog implementation of QCMEC were to do better than even a naive C implementation. However, it is not our goal to compare the efficiency achievable with different programming languages.

> *Page 8, before Th. 4.1: You write "We only show the statement of our soundness and completeness result since a fully worked out proof would require a very detailed account of the semantics of λ-prolog". So why implement QCMEC in λ-prolog at all? By implementing a straightforward QCMEC algorithm using a language with a more intuitive semantics, the soundness/completeness proof would be more or less trivial (once again, compare with your PSPACE-membership proof of QCMEC).*

We strongly disagree. A correctness proof requires showing that the computation induced by the execution of an algorithm achieves the effects dictated by its specification, and nothing else. If we want to be serious about this, we must in particular relate the individual steps in the computation to aspects of the specification. In the case we consider, this is quite immediate since the representation language (λ-Prolog) is a logic, and the specification of QCMEC is given by means of logical definitions. Relating lambda-Prolog derivations (i.e. execution traces) to these specifications is almost trivial; however, a fully formal proof requires examining all possible cases and all possible rules, making it long and tedious. A proof of this kind can be found in [1][[4] (to be published in the Journal of Logic Programming)]. Had we relied on an implementation in an imperative language such as C, we believe that a similar proof, although theoretically possible, is simply not feasible in practice (at least in a reasonable amount of time).

This was the logician's answer. The vast majority of the people who are likely to cast their eyes on our work will be interested in the more intuitive aspect of a correctness proof that we call immediacy: how far apart is the implementation from the formalism it implements? Here, our lambda-Prolog implementation is a direct transliteration of the specification of the semantics of QCMEC. It would not be so had we used C.

> *Chapter 5 "Complexity Analysis" Why not rearrange the section as follows: Begin the section by proving that QCMEC is in PSPACE. Then there is no need for first showing that CMEC is in PSPACE, extending the result to EQCMEC and finally realize that the proof for EQCMEC also works for QCMEC.*

You propose a top-down organization of the chapter; we arranged it in a bottom-up fashion, which gives us for free some of the hardness results (e.g., QCMEC-hardness immediately follows from EQCMEC-hardness). We prefer to keep things as they stand.

[1]Ref: http://www.stanford.edu/~iliano/papers/udmi-37_96.ps.gz

Page 8, col 2, proof of Th. 5.3.: You write "...the algorithm enters in an AND (resp. OR) state. It nondeterministically choose one among α and β and evaluates it in w." This should be rephrased. If you are in an OR state, the sentence is correct but if you are in an AND state, no nondeterministical choice takes place. Instead, the ATM checks that all paths starting in the AND state is an accepting path.

In the paper, we describe an alternating polynomial time algorithm. The algorithm neither checks that all paths starting in an AND state are accepting paths, nor searches for an accepting path starting from an OR state. If it were to, it would not be polynomial time-bounded. In both cases (AND and OR states), the algorithm simply guesses a path and follows it.

By definition, the alternating machine accepts an AND-state configuration (resp., an OR-state configuration) if and only if all (resp., al least one) of its successor configurations are accepting configurations. Thus, the distinction bewteen AND states and OR states comes into play in the definition of acceptance.

It is worth noting that given a problem, a nondeterministic algorithm for this problem does not solve it. Given a "yes" instance of the problem and a succint certificate for it, the nondeterministic algorithm only checks that the instance is indeed a "yes" instance of the problem by exploiting the certificate. For example, in the case of SAT, given a satisfiable propositional formula F and a truth assignment for the variables in F that makes F true, a nondeterministic algorithm for SAT checks that the assignment satisfies F.

Notice also that AP is equal to PSPACE, that is, finding an alternating polynomial time algorithm for a given problem allows us to conclude that the problem can be solved in polynomial space. Thus, it is not surprising that the behaviour of the algorithm in an AND and in an OR state is the same, since the space requirements in both states are the same.

See [12] (chapter 16) for further details.

Page 10, col 1, end of proof of Th. 5.3. Please give the proof of why $w \vdash F_1$ iff G is true. Or, at least, give the reader a hint...

While answering this comment, we noticed a minor imprecision in the definition of the formula defining F_k. The correct formulation is as follows:

$$F\_k = \begin{cases} \Diamond\hat{F} & \textit{if } k = n \\ \Diamond(\psi_{k+1} \wedge \Box\hat{F}) & \textit{if } k = n\text{-}1 \\ \Diamond(\psi_{k+1} \wedge \Box(\psi_{k+2} \rightarrow F_{k+2})) & \textit{otherwise} \end{cases}$$

Coming back to your question, we will now provide further details in order to help the reader to understand the proof.

Let G be a QBF with variables in $V = \{x_1, x_2, ...x_n\}$ and H be an EC-structure as defined in the proof.

A knowledge state w in W_H induces an assignment t_w of the variables in V: for every variable x in V, let $t_w(x)$ be true if and only if $w \models p_x(e_x, e_{\neg x})$. Otherwise, that is, if $w \models \neg p_x(e_x, e_{\neg x})$, then $t_w(x)$ is false. Notice that different knowledge states may induce the same assignment. Indeed, only the relative order of the events e_x and $e_{\neg x}$ for every variable x in V is relevant. This many-to-one correspondence between knowledge states and assignments establishes an equivalence relation on the set of all the knowledge states for H such that the truth of a propositional CMEC-formula on H (i.e. a CMEC-formula on H without modalities) is invariant within each equivalence class.

If F is a propositional formula (the matrix of some QBF G), the corresponding (propositional) CMEC-formula $\hat{F}$ is obtained by replacing every occurrence of the variable x in F with $p_x(e_x, e_{\neg x})$. Given a knowledge state w in W_H, $w \models \hat{F}$ iff t_w satisfies F. Hence, $w \models \Diamond\hat{F}$ (resp. $w \models \Box\hat{F}$) iff F is satisfiable (resp. valid).

In the proof, we exploit this correspondence in order to map the propositional quantifiers of a QBF into the modal operators of a CMEC-formula. However, there is a technical problem with this solution: the scope of a quantifier is limited to the variable that it quantifies, whereas the scope of a modal operator in the formula $\Box F$ or $\Diamond F$ extends to the whole formula F. In the proof, we take advantage of the formulas ψ_k in order to restore the correct context for the evaluation of $\hat{F}$.

We invite the skeptical reader to make some small examples with two or three nested quantifiers.

> *Page 11, col 1: You write "In Section 4 we have transliterated the definition of QCMEC and its subcalculi in the higher-order logic programming λ-prolog. The directness of the implementation allows checking easily that the complexity of the implemented algorithms coincide with the bounds proved in this section for the problems they implement". One should be extremely careful when justifying complexity results with implementations in logic programming languages. Such languages often introduce large overheads and it is difficult to know the exact complexity of the different operations (unification,...) that are performed by the systems. Your assertion may very well be correct, but it needs further justification.*

This observation is true in general of the implementation of any programming language. We do not have enough insight about the internals of the implementation we used to answer it in one way or another. However, because of the relatively simple structure of our programs, we do not expect the overhead introduced by the interpreter we used to alter the bounds we obtained.

Vol. 2: *219*, 30.6.1998
http://www.etaij.org/ra/rac/008/

David Poole:

Decision Theory, the Situation Calculus and Conditional Plans

Abstract of the article

This paper has been submitted to the Electronic Transactions on Artificial Intelligence. For the purpose of the review discussion, it has first been published and posted by Linköping University Electronic Press; it is available at http://www.ep.liu.se/ea/cis/1998/008/

This paper shows how to combine decision theory and logical representations of actions in a manner that seems natural for both. In particular, we assume an axiomatization of the domain in terms of situation calculus, using what is essentially Reiter's solution to the frame problem, in terms of the completion of the axioms defining the state change. Uncertainty is handled in terms of the independent choice logic, which allows for independent choices and a logic program that gives the consequences of the choices. As part of the consequences are a specification of the utility of (final) states, and how (possibly noisy) sensors depend on the state. The robot adopts conditional plans, similar to the GOLOG programming language. Within this logic, we can define the expected utility of a conditional plan, based on the axiomatization of the actions, the sensors and the utility. Sensors can be noisy and actions can be stochastic. The planning problem is to find the plan with the highest expected utility. This representation is related to recent structured representations for partially observable Markov decision processes (POMDPs); here we use stochastic situation calculus rules to specify the state transition function and the reward/value function. Finally we show that with stochastic frame axioms, action representations in probabilistic STRIPS are exponentially larger than using the representation proposed here.

Vol. 2: *220*, 30.6.1998
http://www.etaij.org/ra/rac/009/

Marc Denecker, Daniele Theseider Dupré, and Kristof Van Belleghem:

An Inductive Definition Approach to Ramifications

Abstract of the article

This paper has been submitted to the Electronic Transactions on Artificial Intelligence. For the purpose of the review discussion, it has first been published and posted by Linköping University Electronic Press; it is available at http://www.ep.liu.se/ea/cis/1998/007/

In the current state of the art on the ramification problem, the purpose of causal laws is to restore the integrity of state constraints. In contrast with this view, we argue that causal laws should be seen as representations of how physical (or logical) forces and effects propagate through a dynamic system.

We argue that in order to obtain a natural and modular representation of the effect propagation process, a causal rule language is needed which allows for *recursion* to model effects on mutually dependent fluents, for *negation* to model effects which propagate in the *absence* of other effects, and with *complex fluent formulae* to model causality in a compact and natural way.

A fundamental property of the process of effect propagation in a dynamic system is that it is *constructive*: effects and change propagations do not spontaneously appear without an external cause. To adequately model the constructive nature of the physical change propagation process, we base the semantics of the formalism on the principle of inductive definition, the main mathematical constructive principle. We use a generalised inductive definition principle, generalising Clark completion and circumscription, to define a unique intended semantics for causal theories. Our approach can deal in particular with cyclic dependencies between effects, in such a way that to some definitions, in spite of syntactic cycles, we can assign a unique intended semantics in a constructive way, while other "actually cyclic" definitions are explicitly detected by the semantics as bad definitions.

Our formalism allows to express the effect propagation process with high precision. Evidence for this is found in the fact that in many applications, a representation is obtained which - almost as a side effect - correctly models the interacting effects of simultaneous actions. Our approach is presented independent of a specific time structure, such as situation calculus, $\mathcal{A}$, or Event Calculus, but we briefly discuss how it can be embedded in different time structures.

Vol. 2: *221*, 30.6.1998
http://www.etaij.org/ra/rac/010/

Murray Shanahan:

A Logical Account of the Common Sense Informatic Situation for a Mobile Robot

Abstract of the article

This paper has been submitted to the Electronic Transactions on Artificial Intelligence. It has previously been published in the proceedings of ECAI-96, the European Conference on Artificial Intelligence. For the purpose of the review discussion, the author has posted an electronic copy on his webpage. Please refer to the ETAI webpage structure for access.

Any model of the world a robot constructs on the basis of its sensor data is necessarily both incomplete, due to the robot's limited window on the world, and uncertain, due to sensor and motor noise. This paper proposes a logic-based framework in which such models are constructed through an abductive process whereby sensor data is explained by hypothesising the existence, locations, and shapes of objects. Symbols appearing in the resulting explanations acquire meaning through the theory, and yet are grounded by the robot's interaction with the world. The proposed framework draws on existing logic-based formalisms for representing action, continuous change, space, and shape, but a novel solution to the frame problem is employed. Noise is treated as a kind of non-determinism, and is dealt with by a consistency-based form of abduction.