Electronic Transactions on Artificial Intelligence 39
Vol. 1 (1997): *39-72*
http://www.ep.liu.se/ej/etai/1997/003/

Refereed Article

Antonis Kakas and Rob Miller:

Reasoning about Actions, Narratives, and Ramification

Abstract: *The Language $\mathcal{E}$ is a simple declarative language for describing the effects of action occurrences within a given narrative, using an ontology of actions, time points and fluents (i.e. properties which can change their truth values over time). This paper shows how $\mathcal{E}$ may be extended to deal with ramifications. More precisely, we show how Language $\mathcal{E}$ domain descriptions can include statements describing permanent relationships or constraints between fluents, and how the model theoretic semantics of $\mathcal{E}$ can be extended in an intuitive way to ensure that the effects of actions are appropriately propagated via such statements, whilst retaining $\mathcal{E}$'s simple approach to the frame problem. We also show how Event Calculus style logic programs may be used to compute consequences of such domain descriptions using standard SLDNF, even when only incomplete information is given about some initial state of affairs. Because of $\mathcal{E}$'s generality, these techniques are easily adaptable to other formalisms for reasoning about actions, such as the Language $\mathcal{A}$ and the Situation Calculus.*

Remark: Various Prolog program listings associated with this paper are also available from http://www.dcs.qmw.ac.uk/~rsm/abstract15.html, or by reference from http://www.ep.liu.se/ea/cis/1997/012/ .

Publication and review history

1. First version *published* by Linköping University Electronic Press on 16.10.1997 and permanently available at
   http://www.ep.liu.se/ea/cis/1997/012/
2. *Received* by the research area "Reasoning about Actions and Change" of the Electronic Transactions on Artificial Intelligence, posted on its web site, and advertised in its Newsletter on 16.10.1997; the summary on 22.10.1997.
3. Publicly *discussed* in the ETAI area of Reasoning about Actions and Change during October, 1997 - March, 1998. The discussion protocol is available and will be retained in the Article Interaction Page at
   http://www.ida.liu.se/ext/etai/received/rac/003/aip.html
4. Revised version, with minor corrections compared to the first version, *published* by Linköping University Electronic Press on 16.2.1998 and permanently available at
   http://www.ep.liu.se/ea/cis/1997/012/
   under the label "Revised publication 1998-02-16".
5. Revised version *accepted* after due refereeing and according to scientific journal standards by the Electronic Transactions on Artificial Intelligence (ETAI) on 23.4.1998. Comments by the referees are included in the Article Interaction Page.
6. Revised version appears in Electronic Transactions on Artificial Intellligence, Volume 1 (1997), Issue 4, pages 39–72. The issue and the annual volume are made permanently available at
   http://www.ep.liu.se/ej/etai/1997/
7. Paper editions of the ETAI issue and the ETAI volume containing this artice *republished* by the Royal Swedish Academy of Sciences. For purchase of this edition, please refer to the URL mentioned in the previous item.

The review policy and the quality requirements for acceptance are documented at
http://www.ida.liu.se/ext/etai/info/

Article maintenance

The Electronic Transactions on Artificial Intelligence maintains an Article Interaction Page for this article at
http://www.ida.liu.se/ext/etai/received/rac/003/aip.html
Besides the publication and review history, it contains links for contacting the author(s), as well as to amendments and other related information for the article. It is intended to keep these links up-to-date in the future.

Summary

ETAI authors are recommended that each article be accompanied by a summary. Longer and more specific than a conventional abstract, it should specify in concrete terms what are the new results in the article. If present, the summary also plays a role in the refereeing process: referees are asked to judge whether the results as specified in the summary *are of importance to the field, and whether the body of the article* gives sufficient evidence *for the claims made in the summary. – The Editor.*

This paper shows how the Language $\mathcal{E}$ [Kakas and Miller, 1997] may be extended to deal with ramifications, and how domain descriptions written in this extended language may be translated into Event Calculus style logic programs. These programs are shown to be sound even when only incomplete information is given about some initial state of affairs.

The Language $\mathcal{E}$ was developed partly in response to the Language $\mathcal{A}$ [Gelfond and Lifschitz, 1993], which was introduced as the first in a family of "action description languages". Action description languages (such as $\mathcal{A}$) inherit an ontology from the Situation Calculus, whereas the Language $\mathcal{E}$ inherits its ontology (which includes an independent flow of time) from Kowalski and Sergot's Event Calculus [Kowalski and Sergot, 1986]. $\mathcal{E}$ can therefore be regarded as a basic or kernel "event description language". It was developed in the belief that the use of, and comparison between, different ontologies is important in the study of formal reasoning about actions. The semantics of $\mathcal{E}$, like that of $\mathcal{A}$, is model-theoretic, and divorced from computational considerations.

The Basic Language $\mathcal{E}$

The paper begins by reviewing the basic Language $\mathcal{E}$. $\mathcal{E}$'s vocabulary includes a set of fluent constants, a set of action constants, and a partially ordered set of time-points. Basic Language $\mathcal{E}$ domain descriptions can include three types of propositions: t-propositions ("t" for "time point"), h-propositions ("h" for "happens"), and c-propositions ("c" for "causes"). For example, the following domain description (about taking a photograph) contains 1 t-proposition, 3 h-propositions and 2 c-propositions:

```
NOT Picture holds-at 1
Load happens-at 2
Look happens-at 5
Take happens-at 8
Load initiates Loaded
Take initiates Picture when {Loaded}
```

The model theoretic semantics of $\mathcal{E}$ ensures that (for example) this domain description entails the t-proposition

```
Picture holds-at 10
```

The notions of an "initiation point" and a "termination point" are central to $\mathcal{E}$'s semantics. For example, in all models of the above domain, 2 is an initiation point for `Load` and 8 is an initiation point for `Picture`. Time can be discrete or continuous, and need not be linear. Indeed, as a special case time may be modelled as a branching structure of sequences of action constants. This allows the "simulation" in $\mathcal{E}$ of the Language $\mathcal{A}$, by writing and reasoning about t-propositions such as

```
Picture holds-at [Load, Look, Take]
```

Describing Indirect Effects in $\mathcal{E}$

The remainder of the paper discusses an extension of $\mathcal{E}$ to include a fourth type of statement called an r-proposition ("r" for "ramification"). R-propositions express permanent constraints or relationships between fluents. In formalisms which allow for such statements, the effects of actions may sometimes be propagated via groups of these constraints. This gives rise to the "ramification problem", i.e. the problem of adequately and succinctly describing these propagations of effects whilst retaining a solution to the frame problem.

R-propositions are statements of the form

```
L whenever {L1, ..., Ln}
```

The intended meaning of this statement is "at every time-point that L1, ..., Ln, L holds, and hence every action occurrence that brings about L1, ..., Ln also brings about L". Hence the semantics of $\mathcal{E}$ is extended so that from a static point of view r-propositions behave like classical constraints, but they are unidirectional in terms of the way they propagate temporal change initiated by action occurrences. This is achieved in the main by appropriately extending the definitions of an initiation and a termination point. These definitions are now recursive, or in terms of least fixed points. R-propositions thus provide a simple, succinct method of expressing domain constraints, and the corresponding semantics behaves in a satisfactory way for a range of examples found in the literature.

The use of r-propositions is illustrated in the paper with two examples. The second of these is Thielscher's electric circuit example [Thielscher, 1997]. This example is of interest because it presents difficulties for what Thielscher describes as "categorisation-based" approaches to ramification. In the (Language $\mathcal{E}$ version of the) example, the permanent configuration and dynamic behaviour of an electric circuit is described by r- and c-propositions such as

```
Light whenever {Switch1, Switch2}
Relay whenever {Switch1, Switch2}
-Switch2 whenever {Relay}
CloseSwitch1 initiates Switch1
OpenSwitch1 terminates Switch1
CloseSwitch2 initiates Switch2 when {-Relay}
```

If `Switch2` already holds (i.e. switch number 2 is connected) and a `CloseSwitch1` action occurs, say at time-point T1, the extended semantics of $\mathcal{E}$ ensures that (in all models) the effect of this event is propagated through the first of the r-propositions above, so that `Light` becomes true. This is because the least fixed point definition of an initiation point ensures that T1 is an initiation-point for `Switch1`, and hence (by the recursive definition) an initiation point for `Light` by the first r-proposition above.

Logic Program Translations

The paper gives a translation method from $\mathcal{E}$ domain descriptions into logic programs, and gives a proof of the correctness of the translation (as regards derivation of t-propositions) for a wide class of domains. As in [Kakas and Miller, 1997], over-zealous application of logic programming's closed world assumption is avoided by representing negated fluents inside a meta-level `HoldsAt` predicate. For example, `Relay holds-at 2` is translated as `HoldsAt(Pos(Relay),2)` and `-Relay holds-at 2` is translated as `HoldsAt(Neg(Relay),2)`. C-propositions such as

```
CloseSwitch2 initiates Switch2 when {-Relay}
```

are translated into two clauses:

```
Initiates(CloseSwitch2,Switch2,t) <-
    HoldsAt(Neg(Relay),t).
PossiblyInitiates(CloseSwitch2,Switch2,t) <-
    not HoldsAt(Pos(Relay),t).
```

The first of these clauses gets used to compute changes in the truth value of `Switch2` and other fluents via occurrences of `CloseSwitch2`. The second gets used in the computation of persistence of truth values. (Similar techniques are used in a number of other logic program translations of action formalisms.)

A soundness property is proved for the logic program translations of a general class of domain descriptions, which may include r-propositions. It is stated in terms of SLDNF-derivability: if there is a finite SLDNF derivation of `HoldsAt(Pos(F),T)` (respectively `HoldsAt(Neg(F),T)`) from the program, then `F holds-at T` (respectively `-F holds-at T`) is entailed from the original domain description.

For the examples given in the paper, these logic programs are more-or-less directly executable in Prolog. The relevant Prolog listings are available at `http://www.dcs.qmw.ac.uk/~rsm/abstract15.html`.