The Language $\mathcal{EC}+$

Robert Craven and Marek Sergot

Department of Computing,
Imperial College London,
180 Queen's Gate,
London SW7 2BZ

{rac101,mjs}@doc.ic.ac.uk

Abstract

We present $\mathcal{EC}+$, a reasoning system closely based on the action language $\mathcal{C}+$ of Giunchiglia et al. (2004). Linguistically, $\mathcal{EC}+$ is a subset of $\mathcal{C}+$, retaining the graphical semantics of the latter whilst restricting somewhat the syntax of permissible causal laws. However, the implementation of $\mathcal{EC}+$ is entirely new, using a logic program inspired by the Event Calculus in order to compute the answers to individual queries in an efficent way, depending only on information relevant to the query. There are substantial gains in efficiency, as well as the advantages afforded by a 'trace' mechanism for calculating the reasons for fluents' holding. The relationship between $\mathcal{EC}+$ and the Event Calculus allows us to see the former as an updated version of the latter, providing new language features and a semantics of labelled transition-systems.

The action language $\mathcal{C}+$ of Giunchiglia et al. (2004) is used to specify how properties of systems change over time, and has built-in, high-level support for features of domains such as the ramifications of actions, the default effects of actions, non-determinism, multi-valued fluents, and a nuanced control over inertia. Another very useful feature of the language is semantic: sets of laws of $\mathcal{C}+$, known as 'action descriptions', define labelled transition-systems, graphical structures which provide a bridge to many other formalisms in Computing and AI. (We have exploited this bridge in other work, performing model-checking on finite-state machines defined using $\mathcal{C}+$ action descriptions.)

An implementation exists (the system CCALC[1]) which finds runs of given length through those transition-systems, based on an underlying algorithm of propositional satisfaction. Typically, the user specifies a number of actions which are to occur at given times in a run, and fluents which must be true at given times; CCALC then finds the set of runs which are consistent with the information supplied. A severe disadvantage of the way in which the current implementation of $\mathcal{C}+$ depends on propositional SAT-solvers is that, to answer queries about whether a given fluent is true at a given state along a run, CCALC must construct the entire run, in effect answering every single query which could be posed about the run.

We present $\mathcal{EC}+$, an alternative implementation route for $\mathcal{C}+$ which removes the mechanism of propositional satisfaction, replacing it with a logic program which only considers information relevant to the value of the fluent which the user is querying. $\mathcal{EC}+$ does not need to construct the entire run to answer an individual query, but looks only at those actions and fluents which may have affected the value of the fluent in which the user is interested. The approach is inspired by the Event Calculus (see Kowalski and Sergot (1986)): $\mathcal{EC}+$ comprises, in addition to the laws of the particular action description, a number of common axioms whose form is closely related to the Event Calculus.

Syntactically, $\mathcal{EC}+$ is a subset of $\mathcal{C}+$, restricting the form of laws allowed in action descriptions. The restrictions are mostly designed to exclude the possibility of non-determinism in our systems; aside from that exclusion, most other features of $\mathcal{C}+$ are retained, ensuring that $\mathcal{EC}+$ is an expressive formalism capable of representing many standard domains in AI and knowledge representation. The semantics of $\mathcal{EC}+$ action descriptions are just the same as for $\mathcal{C}+$: we still have the highly useful labelled transition-systems.

When working with $\mathcal{EC}+$, the user typically supplies information about an initial state of the system, and about which actions have occurred at which times of a run. Then queries can be made, of what the value of a given (multi-valued) fluent is at any time along the run. The logic program at the heart of $\mathcal{EC}+$ checks to see whether actions have been performed which are relevant to the determination of the fluent's value, looks at the possiblity of a value for the fluent carrying through by inertia, and considers whether the fluent will take a specified default value. The system we have written in PROLOG also performs preliminary checks on the consistency of action descriptions and information about a run which the user supplies.

Further features of our system for $\mathcal{EC}+$ include a trace facility, which provides an explanation of why a given fluent takes the value it does at a given time. For instance, if a *detonated* fluent is true, the system may explain that this has been so for the last five time-steps, because five steps ago, an *explode* action occurred, whose

[1] See http://www.cs.utexas.edu/users/tag/cc/.

preconditions were that the trip wire was *connected* and the bomb correctly *made*; if desired, the trace may add that the wire was *connected* because we connected it two steps previously, and so on. These causal explanations can be given to an arbitrary degree of nesting, as far back in the past as is desired and possible.

One way of seeing $\mathcal{EC}+$ is as an alternative implementation route for $\mathcal{C}+$, exploiting the features of a top-down, logic-programming approach which considers only the information relevant to the determination of the value of a fluent. Yet our work can also be seen as an updated version of the Event Calculus, providing a graphical semantics to a version of that formalism which includes intuitive support for ramifications, multi-valued signatures, and a precise formulation of interactions amongst different forms of default and inertia. This also means that $\mathcal{EC}+$ forms part of a much larger project of connecting and comparing different formalisms for reasoning about action and change.

References

E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004.

R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.