Linköping Electronic Articles in
Computer and Information Science
Vol. 3(1998): nr 17

Cognitive Robotics Logic and its Metatheory:

Features and Fluents Revisited

Erik Sandewall

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1998/017/

Published on October 15, 1998 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in
Computer and Information Science, Vol. 3(1998): nr 17.
http://www.ep.liu.se/ea/cis/1998/017/. *October 15, 1998.*

This URL will also contain a link to the author's home page.

Abstract

Cognitive Robotics Logic (CRL) is an extensible logic language for characterizing actions and change, in particular for use in cognitive robotics. Its development emphasizes the issues of syntax, expressivity, underlying semantics and entailment methods (defined in terms of the semantics). Development of proof methods is de-emphasized. The salient results from this approach refer to the range of applicability and other related properties of the entailment methods. These results constitute a metatheory of actions and change.

CRL is syntactically defined as a base language and a surface language. The base language is characterized by the following aspects:

- its three major predicates *Holds*, *Occurs*, and *Occlude*, which provide coherence when the language is extended;
- its reportoire of categorial functions, which is augmented when additional expressiveness is required in the language.

The surface language provides additional notational convenience, and is defined by translation to the base language.

The range of expressivity includes actions with duration, nondeterministic actions, actions in hybrid worlds with piecewise continuous fluents, some forms of ramification and causation, imprecise sensors and actuators, action failure, and some aspects of goal-directed agent behavior.

Entailment methods are functions that map scenario descriptions to sets of *intended models*. They are defined using a reportoire of set-theoretic operations on sets of formulas and sets of models, including but not restricted to minimizing a set of models with respect to a preference relation.

A progression of underlying semantics is defined, beginning with the partial state-transition semantics and its immediate generalization, the trajectory semantics. These underlying semantics are used for the formal analysis of the range of applicability of various entailment methods, including both those proposed by the others in this research, and those that developed in the course of the present work.

1 Scope

Contemporary logics of actions and change provide answers to questions of *expressivity* and *implementation*. Logics are required to express simple scenarios correctly. They are also expected to have a proof theory, for example using a reduction to classical (monotonic) first-order logic, or by relying on logic programming. Examples of such logics of actions and change include several variants of the Situation Calculus [j-aij-64-337] and of action description languages [j-jlp-17-301], the Event Calculus [j-aij-77-249], and Time and Action Logic [f-linep-98-15].

The approach and results presented here complement those efforts by providing a *metatheory* of actions and change whose first purpose is to relate logics of actions and change, such as those mentioned above, to their intended meaning. For example, the metatheory provides answers for questions of the form "under which conditions is a specific logic **L** guaranteed to give the intended results for given queries?". The "intended results" are then defined in terms of an *underlying semantics*; the definition and the use of such an underlying semantics is a characteristic feature of the metatheory.

Logics of actions and change have two major application areas: for commonsense reasoning, and for cognitive robotics. We emphasize the latter aspect, which has led us to address the modelling of continuous processes, the actual properties of sensors and actuators, and the modelling of the autonomous agent itself. We therefore use the term **Agent Systems Metatheory** and the acronym ASM for characterizing the whole approach.

1.1 Cognitive Robotics Logic

Concrete results on the metatheory level are often best formulated in terms of a particular logic syntax. However the results are not tied to that choice of syntax, since results formulated for one logic can often be mapped to corresponding results for another logic of actions and change. In the ASM research we have used roughly the same logic syntax over a period of almost ten years, but without emphasizing any particular name for it. We have also chosen to allow minor variations in the logic between different articles. It turns out, however, that for the purpose of reference it is useful to have a stable definition and a specific name. The present article provides an encompassing definition of the logic and introduces the name **Cognitive Robotics Logic** for it, with the acronym CRL.

This Cognitive Robotics Logic provides a representational framework *within which* various subsets can be identified for different purposes. Sometimes a sublanguage is *selected* in order to correspond to a particular need. At other times, a sublanguage is *identified* as a language domain within which a particular property can be proven to hold. In both cases, it is useful to have a broad and coherent representational system to draw from, *although* there has not yet been any occasion where the whole language was used at once. In this respect the language used by a metatheory, such as the CRL, differs from the syntax used in a particular object-level investigation.

For simplicity and uniformity, CRL is formulated as a first-order logic. It inherits the conventional Tarskian semantics, which is complemented by the underlying semantics in ways that will be further described below.

To summarize: the Agent Systems Metatheory (ASM) is able to make statements about range of applicability and related properties in various logics of actions and change. We define a logic in the restricted sense of a first-order syntax for logic formulae, called Cognitive Robotics Logic (CRL), and use it for the primary formulation of ASM results. From there, the results can be translated to apply to other logics for actions and change.

The use of CRL in ASM is strongly semantically oriented: it provides notation, semantics, and semantics-based properties, but says nothing about proofs. Besides for assessment results, this line of research has made several contributions with respect to expressivity, such as the use of *occlusion* and of *filtering*. These developments have later been adopted by several of the specific logics of actions and change.

Although CRL itself says nothing about proofs, some of the related research does. CRL was used (before its present name was assigned to it) by Patrick Doherty and his students as the basis for **Time and Action Logic** (TAL), which is a regular logic of actions and change that provides well founded answers to issues of expressivity (in particular for ramification, concurrency, and delayed effects) and of proof methods using reductions to first-order logic. The relationship between TAL and CRL are further described below and in Doherty's concurrent article about TAL.

CRL provides expressivity for the following three broad areas:

1. Classical issues in reasoning about actions and change: ramification, qualification, and concurrency. These parts have largely been developed in the framework of TAL.
2. Hybrid systems: mixed continuous and discrete change; description of actions on multiple levels of details; characterization of sensors and actuators.
3. Modelling of the behavior of the cognitive robot or agent itself.

The present article is intended as a reference article, meaning that it summarizes the common core of previously published work, and at the same time it adopts a cleaner and more uniform notation. The notation has been chosen so as to conform to TAL in the relevant parts. Forthcoming articles will be able to refer to the present one for background and basic definitions, instead of repeating them each time. Because this is the goal, the article intentionally only describes previously published approaches and results. It differs from a tutorial in that it assumes a general familiarity with the topic of research "reasoning about actions and change". Please refer to our article in the Handbook of Logic in AI, [s-Gabbay-94-439] for a general introduction to this topic.

1.2 Meta-level elaboration tolerance

McCarthy has introduced the concept of *elaboration tolerance* [c-fcs-98-198]. On the object level, a logic is elaboration tolerant to the extent that it allows scenario descriptions to be modified and extended with additional information of new kinds. We propose that elaboration tolerance is also an important property on the meta level: a theory of actions and change is *meta level elaboration tolerant* to the extent that it can readily be extended with constructs for additional types of phenomena.

Meta-level elaboration tolerance is always a useful property, but it is absolutely essential when choosing a logic framework for the Agent Systems Metatheory. CRL has therefore been designed as an *extensible language* in the sense that there are well defined ways for extending its expressiveness with respect to both the syntax and the semantics, and in such a way that proven results from earlier language generations can be inherited in controlled ways by later generations. The present article describes those extension mechanisms. Several of the fundamental features of CRL have remained unchanged for a period of ten years, while the expressiveness has been extended considerably.

2 Syntax of simple CRL

Full CRL allows expressions for fluents, actions, and fluent values to be formed using functions with objects as arguments, for example the fluent $colorof(car4)$ and the action $paint(car4, red)$. The use of such functions is very important in most applications, but when it comes to defining the syntax and the properties of the language it is mostly a distraction: it increases the complexity of the definitions, but without introducing any significant new issues. Therefore, we shall first introduce the subset *simple CRL* where this possibility is excluded. The extension to *CRL with domain objects* will be discussed in section 6.

2.1 Surface language and base language

One of CRL's extension mechanisms is to use two levels of language: a *surface language* and a *base language*. The base language is a quite conventional first-order logic with a minimal number of constructs, which is convenient for the analysis of formal properties of the language and, in particular, the reasoning methods using the language. The surface language has a larger set of constructs, but all of them are defined in terms of translations to the base language. The surface language is designed in order to provide the best possible expressiveness for actual scenario descriptions in concrete applications.

Additional expressivity has often been obtained by adding constructs to the surface language, but it has been possible to define them in terms of the same base language as before. This has assured the survival and continued applicability of theoretical results inasmuch as they are formulated in terms of the base language.

The constructs in the surface language are of three kinds:

- *Core constructs*, that is, constructs that have immediate counterparts in the base language.
- *Abbreviations*, that is, constructs that can be translated to the base language without recourse to any information outside the abbreviation construct itself.
- *Context dependent constructs*, which can only be translated using other formulas in the same scenario description (set of axioms).

Several of our articles have treated the base language and the core constructs of the surface language as identical. TAL introduced the use of separate syntax for those two, and we follow that practice in CRL. We now proceed to the definition of CRL syntax.

2.2 Types and predicates

Simple CRL uses the following four types:

- *timepoints*, usually chosen as the integers or as the real numbers. The word *time* is used as synonym of timepoint.

- *fluent names* or *features*, formed constructively by starting with a finite number of individuallly named objects, and extending the domain using *categorial functions* (see below).
- *fluent values*, which for the purpose of simple CRL are chosen as the integers or the reals. The reals are used when continuous valued fluents are needed. Otherwise the integers are used as a universal fluent-value domain, avoiding the need to have different value domains for different fluents.
- *actions* which, like fluents, are formed from a finite number of individually named objects, extended using categorial functions.

The distinction between fluent and fluent name is similar to the distinction between function and function symbol. Technically speaking, a function is a set of argument-value pairs, but we often say 'function' when we mean 'function symbol'. Likewise, a fluent is really a function from features to fluent values, and a feature is the same as a fluent name. However, in common parlance the word 'fluent' is often used to mean 'fluent name', and we will sometimes adopt that (mal-)practice in the present text.

Variables for these types are written as single letters, whereas constants are written as words of more than one letter. Usually, variables for timepoints are written using the letters s and t, fluent names using the letter f and occasionally d or b, fluent values using v or adjacent letters in the alphabet, and actions using a and sometimes e and g (e for event, g for goal).

There are three predicates, which are written as follows in the CRL base language.

- $Holds(t, f, v)$, expressing that the fluent name f has the value v at the time t.
- $Occurs(s, t, a)$, expressing that the action a occurs with the starting time s and the termination time t.
- $Occlude(t, f)$, expressing that the fluent name f is occluded at the time t, which means that at that time it is exempt from the usual "frame" assumptions of persistence or inertia.

The *Holds* predicate is of course a standard used in most approaches in this field, and the *Occurs* predicate is also quite obvious. The *Occlude* predicate was introduced by us in 1989 [c-ijcai-89-894] with the name X, to be used together with *Holds* and *Occurs*. The notation has been stable since then, with only minor variations. Successive extensions to the surface language have often been accomodated by defining their translations to this fixed set of predicates in the base language. We therefore consider them to play a fundamental role in CRL.

The corresponding constructs in the core of the surface language are as follows. Different authors and different articles have used slightly different notations, so we offer several alternatives here:

- $Holds(t, f, v)$ may be written $[t] f \hat{=} v$ or $\mathsf{H}(t, f : v)$ in the surface language.
- $Occurs(s, t, a)$ may be written $[s, t] a$ or $\mathsf{D}([s, t], a)$ in the surface language. (Here, D stands for 'do').
- $Occlude(t, f)$ may be written $\mathsf{X}([t], f)$ in the surface language.

For the purposes of some works, it has not been necessary to include *Occurs* in the base language, and the corresponding surface-language expression $[s,t]a$ has been considered as a context dependent construct, translated in each case to an expression formed using *Holds* and *Occlude*.

2.3 Categorial functions

The types of fluents and actions are categorial, in the sense that they represent the categories in which the world is modelled in this logic. By a *categorial function* we mean a function that maps fluents and/or actions to new fluents and/or actions. To be precise, the arguments and values are *fluent names*, i.e. features as remarked above, or similarly for *action names*.

While the three predicates *Holds*, *Occurs*, and *Occlude* are being kept fixed, it is often possible to add more expressivity by introducing additional categorial functions. (This is one of CRL's extension strategies). The following are the categorial functions that have been introduced and used so far. Notice, however, that not all the articles use all of them: usually only some of the categorial functions are considered at a time.

2.3.1 Categorial functions for use with continuous time

In [c-kr-89-412] we introduced the idea of embedding differential calculus in logic for the purpose of characterizing *hybrid systems*, that is, systems exhibiting real-valued and piecewise continuous fluents over continuous time. This required the following operators:

- If f is a continuous valued fluent, then ∂f is a fluent whose value at time t is the derivative of the value of f at time t provided that f has a derivative there.
- If f is any fluent, then λf is the left limit value of f and ϱf is similarly the right limit value of f in case those are well defined.

In those cases where the derivative, left limit value, or right limit value are not defined from the fluent f, it remains a modelling decision whether to assign a specific value to ∂f, λf, and ϱf, respectively. This requires additional axioms in each case.

It follows that if f is continuous at time t then f, λf, and ϱf have the same values there. On the other hand, suppose for example that f represents the horizontal velocity of a ball that bounces without loss of energy against a vertical obstacle at time t. The resulting constraint is expressed as follows:

$$Holds(t, \lambda f, v) \wedge Holds(t, \varrho f, v') \rightarrow v = \text{-}v'$$

2.3.2 Composition of actions

Composite actions are written using the following operators, which were introduced and used for assessments in [b-Sandewall-94], chapter 12:

- ; for sequential composition of actions,
- if ... then ... else ... for conditional composition,
- do ... until ... for repetition.

For some purposes it is possible to view these operators as abbreviations, so that they only exist in the surface language.

If p is a propositional fluent, then $test(p)$ is an action that is always applicable, executes instantly, does not change the state of the world, and succeeds at time t iff p is true at that time. Such actions are used as elements in composite actions.

2.3.3 Categorial functions for sensors and actuators

If f is a fluent representing the true value of a quantity in the environment, we let $obs(f)$ represent the observed value of f according to the available information source(s). Also, if f is a fluent whose value can be controlled using the robot's actuators, we let $set(f)$ represent the value that the actuators are requested to set.

For example, knowledge about the accuracy of the sensor or information source for f may be stated by an axiom such as

$$Holds(t, f, v) \wedge Holds(t, obs(f), v + d) \rightarrow \mid d \mid \leq 0.2$$

A control rule saying that if the distance to the car in front of 'me' is less than 10 meters, then the velocity $vel(me)$ of our car shall be 2 m/s less than the velocity of the car in front might be stated as

$$Holds(t, dist(me, CarInFrontOf(me)), d) \wedge d < 10 \wedge$$
$$Holds(t, vel(CarInFrontOf(me)), v) \rightarrow Holds(t, vel(me), v\text{-}2)$$

Drakengren introduced a variant of *Holds* that referred to observation fluents $obs(f)$. The present notation is immediately intertranslatable with his proposal.

2.3.4 Categorial functions for applicability, invocation, and failure of actions

If a is an action, then we let $app(a)$ be the fluent that is true at a time t iff a is applicable at that time. Also, $inv(a)$ is momentarily true at t iff a is invoked then. Finally, $fail(a)$ is momentarily true at t iff a terminates with failure at that point in time.

The article [j-etai-1-105] is the latest source for axioms characterizing the properties of these categorial functions.

2.3.5 Formation of propositional fluents

Fluents intended to have a truthvalue are represented with the values 1 and 0 in the base language. In the surface language, these are written as true and false , respectively.

If f is a fluent and v is a value, then $f : v$ is the fluent that is true iff f has the value v. Thus

$$Holds(t, f : v, \textsf{ true }) \leftrightarrow Holds(t, f, v)$$

An expression $\lambda f : v$ is parsed as $(\lambda f) : v$.

2.3.6 Uses of these categorial functions

In [j-etai-1-105] and [c-kr-98-304] the categorial functions for applicability, invocation, and failure were used for characterizing goal-directed behavior.

An earlier article, [j-aicom-9-214], [c-hart-97-3], used similar notations for relating high-level and low-level descriptions of actions. (The notation used in that article is now viewed as surface constructs that can be reduced to the categorial functions that have been introduced in this section).

2.4 Branching time

Although CRL is mostly used with linear time, the definitions in [b-Sandewall-94] use a general framework for the time domain that subsumes both linear time and branching time. The assessment proofs are formulated for linear time, but there is a concise description of how the definitions and proofs can be generalized to the case of branching time. It seems clear that all the assessment results generalize without additional restrictions. The specifics are found in sections 6.1.8, 6.6.3, 8.5, 9.5, and 10.4 of the book.

2.5 Abbreviations

The following abbreviations provide additional convenience in using the language.

2.5.1 Distributive abbreviations

The core constructs for *Holds* and *Occlude* are generalized as abbreviations so that they can be written for intervals instead of single timepoints:

- $[s,t]f \hat{=} v$ means that $[u]f \hat{=} v$ holds for every timepoint u in the closed interval $[s,t]$.
- $\mathsf{X}([s,t], f)$ similarly.
- Open and semi-open intervals are allowed similarly.

Also, the constructs based on the *Holds* predicate are generalized as follows with respect to the fluent-value combination:

- $[t]p$ abbreviates $[t]p \hat{=}$ true for propositional fluents.
- $[t]p \wedge q$ means that both p and q have the value true at time t, where p and q are propositional fluents. Similarly for the other propositional connectives, such as $\vee$ and $\neg$.
- Expressions such as $[t]f : v \wedge f' : v'$ are then also well defined.
- $[t]f = d$ means that the values of the fluents f and d at time t are equal, and similarly for other relations besides equality ($<$, $\leq$, etc).
- Expressions formed using arithmetic operations on fluent values are interpreted in the obvious way, as in for example

  $$[t]f_1 + f_2 \leq f_3$$

When intervals are used together with composite fluent-value expressions, the latter are the "inner" ones. Thus $[s,t]p \vee q$ means only that $p \vee q$ holds at each timepoint in the interval $[s,t]$.

All the abbreviations of the form $[t]\varphi$ can also be written $\mathsf{H}(t, \varphi)$.

2.5.2 Referring to change

The following abbreviation constructs are used in TAL for characterizing change over discrete time. We generalize them here so that they apply to continuous time as well.

- $\mathsf{C}_T([t], p)$ where p is a 'propositional' fluent and for discrete time abbreviates $[t\text{-}1]\neg p \wedge [t]p$. For continuous time it abbreviates $[t]\neg\lambda p \wedge \varrho p$. In either case it expresses that p becomes true at time t.

- $\mathsf{C}_F(\texttt{[}t\texttt{]}, p)$ where p is a 'propositional' fluent and for discrete time abbreviates $\texttt{[}t\text{-}1\texttt{]}\, p \wedge \texttt{[}t\texttt{]} \neg p$. For continuous time it abbreviates $\texttt{[}t\texttt{]}\, \lambda p \wedge \neg \varrho p$. In either case it expresses that p becomes false at time t.

We also generalize to the following:

- $\mathsf{C}(\texttt{[}t\texttt{]}, f, v)$ for discrete time abbreviates $\texttt{[}t\text{-}1\texttt{]} \neg (f : v) \wedge \texttt{[}t\texttt{]}\, f : v$
- $\mathsf{C}(\texttt{[}t\texttt{]}, f, v)$ for continuous time abbreviates $\texttt{[}t\texttt{]} \neg (\lambda f : v) \wedge \varrho f : v$

In all cases the first argument may be written t in the place of $\texttt{[}t\texttt{]}$.

2.5.3 Referring to the execution of actions

The categorial functions defined in 2.3.4 are the primitive ones for characterizing the execution of actions, together of course with the predicate *Occurs*. The following notations, now viewed as abbreviations, were introduced in [j-aicom-9-214]:

- $\mathsf{G}(s, a)$ expresses that the action a is invoked ("go") at time s. It is defined as $\mathsf{H}(s, inv(a))$.
- $\mathsf{A}(s, a)$ expresses that the action a is applicable at time s. It is defined as $\mathsf{H}(s, app(a))$.
- $\mathsf{D}_{\mathrm{s}}(\texttt{[}s, t\texttt{]}, a)$ expresses that the action a is executed and succeeds over the time interval $\texttt{[}s, t\texttt{]}$. It is defined as

  $$\mathsf{D}(\texttt{[}s, t\texttt{]}, a) \wedge \neg \mathsf{H}(t, fail(a))$$

- $\mathsf{D}_{\mathrm{f}}(\texttt{[}s, t\texttt{]}, a)$ expresses that the action a is executed and fails over the time interval $\texttt{[}s, t\texttt{]}$. It is defined as

  $$\mathsf{D}(\texttt{[}s, t\texttt{]}, a) \wedge \mathsf{H}(t, fail(a))$$

- $\mathsf{D}_{\mathrm{c}}(\texttt{[}s, t\texttt{]}, a)$ expresses that the action a started executing at time s and is still in progress at time t. It is defined as

  $$\exists u [\mathsf{D}(\texttt{[}s, u\texttt{]}, a) \wedge t \leq u]$$

  The index c stands for "continues".

2.5.4 Constructs defined in terms of occlusion

The occlusion concept has turned out to be surprisingly versatile for characterizing various phenomena. New uses of occlusion are often manifested with new surface-language constructs. The following have been defined.

- $\texttt{[}s, t\texttt{]}\, f := v$ abbreviates $\mathsf{X}((s, t\texttt{]}, f) \wedge \texttt{[}t\texttt{]}\, f \mathrel{\hat{=}} v$. This says that f receives the value v some time during the interval from s to t, so that it certainly has that value at time t. Technically, f is occluded in the interior of the interval from s to t and at time t, but not necessarily at time s. - This abbreviation is regularly used in action laws for actions with duration, where the effect of the action takes place some time during its execution interval. It was first introduced in [c-ismis-93-558].
- $\texttt{[}t\texttt{]}\, \phi \gg \texttt{[}t'\texttt{]}\, \psi$ is used in TAL to express that ϕ becoming true at time t causes ψ to become true at time t'. Technically, it is defined here for the case of $\texttt{[}t\texttt{]}\, \phi \gg \texttt{[}t'\texttt{]}\, f := v$, where it abbreviates

  $$(\texttt{[}t\texttt{]}\, \phi \rightarrow \texttt{[}t'\texttt{]}\, f := v) \wedge (ind(\mathsf{C}, T)(t, p) \rightarrow \mathsf{X}(t', f)) \wedge (t' \geq t)$$

- For the I operator, please see the corresponding article on TAL.

For example, a situation-action rule for direct response can be written on the form

$$[t]\, obs(f) : v \rightarrow set(f') := v'$$

asserting that if the fluent f is observed to have the value v, then the fluent f' shall be set to have the value v'. Equivalently, it can be written

$$[t]\, obs(f) : v \gg [t]\, set(f') := v'$$

This immediately suggests how the rule can be written if a certain delay is allowed between stimulus and response.

2.5.5 Defining the range of fluents

Technically, fluents can only be of two types: integer-valued or real-valued ones. Integers (restricted to non-negative integers) and reals are considered as two disjoint domains; their union is the domain of possible fluent values. The predicate $Discrete(v)$ says that v is a (non-negative) integer.

In practice, many discrete-valued fluents will have values that are understood as qualitative values, such as "having the color green" or "being at or near position L". Such fluent values are technically represented as integers in a finite range $[0, n]$ specified by an axiom. We use the abbreviation $Range(f, n)$ for

$$Holds(t, f, v) \rightarrow Discrete(v) \wedge 0 \leq v \leq n$$

Propositional fluents p (compare subsection 2.3.5) are therefore characterized by $Range(p, 1)$. We let true be an abbreviation for 1 and false for 0. Clearly we have $Range(f : v, 1)$ for all choices of f and v.

2.5.6 Characterization of fluent properties

In the simplest case of using this logic, all fluents are persistent except during the interval of executing an action affecting the value of the fluent. For domains with more complex persistence rules we use the following operations.

- $\mathsf{Per}(f)$ abbreviates $\forall t, v[\mathsf{C}(t, f, v) \rightarrow Occlude(t, f)]$. It indicates that a fluent is persistent, that is, that it does not change or have a discontinuity unless it is occluded.
- $\mathsf{Dur}(f, v)$ abbreviates $\forall t[\neg Occlude(t, f) \rightarrow Holds(t, f, v)]$. It indicates that v is the default value of the fluent f, so unless it is occluded at time t its value at t is v. Thus, the way to specify that f has a nonstandard value at t is to state that it is occluded and what its current value is.

These operations were introduced for TAL and in [f-linep-97-14] as independent predicates. In CRL they are viewed as abbreviations.

2.6 Context-dependent surface constructs

For many purposes, the relation *Occurs* in the base language does not have to be used at all, and action laws are considered to define the translation of action statements in the surface language. This is best explained by an example. Consider the occurrence statement

$$[4, 6]\, Fire$$

and the action law

$$[s, t]\, Fire \leadsto ([s]\, loaded \rightarrow [s, t]\, alive := \textsf{false})$$

In the presence of this action law, the occurrence statement is translated to

$$[4]\, loaded \rightarrow [4, 6]\, alive := \textsf{false}$$

which in turn is further translated to

$$Holds(4, loaded, \textsf{true}) \rightarrow$$
$$\forall u[4 < u \leq 6 \rightarrow Occlude(u, alive)] \wedge Holds(6, alive, \textsf{false})$$

by expansion of abbreviations. This technique avoids the need to minimize the occurrences of actions, which simplifies the formal treatment. However, for some of the more advanced classes of scenarios it doesn't work, namely those where occurences of actions are implied from other known facts.

Notice how the *abbreviations* that were defined in section 2.5 have the same expansion independently of the context where they occur, whereas action expressions using the relation *Occurs* (or its abbreviations) are expanded using an action law that is also found in the scenario description.

For those purposes where it is not sufficient to treat *Occurs* as an abbreviation, one simply has to replace $\leadsto$ by material implication in all the action laws. However, the entailment criteria are then also affected, because one has to restrict action occurrences e.g. by minimizing *Occurs*.

2.7 Scenario descriptions

Several entailment methods and underlying semantics require that scenario descriptions are *partitioned* so that each partition is a set of logic formulae. Different partitions may use different subsets of the full logic language, and they are also treated differently by the entailment methods. Formally, therefore, an *n-place scenario description* is an n-tuple of sets of formulae. The specification of the number of partitions and their syntactic restrictions is done separately in each underlying semantics.

3 Representational capabilities

The base language defined above allows the expression of the following phenomena:

1. Actions with duration (intrinsic in the language)
2. Composite actions
3. Nondeterministic outcome of actions (expressed using the relation *Occlude*)
4. Nondeterministic timing of actions and their effects (also expressed using the relation *Occlude*)

5. Metric time (intrinsic in the language)
6. Continuous time and piecewise continuous fluents (intrinsic in the language)
7. Derivatives of continuous valued fluents (expressed using the operator ∂)
8. Discontinuities, caused by other discontinuities or by some fluent value passing a threshold (expressed using the operators λ and ϱ)
9. Causation-type ramifications (expressed using the relation *Occlude* and the abbreviation $\gg$)
10. Delayed effects (expressed using the operator $\gg$)
11. Interactions in (delayed) effects using influences (expressed using the operator I).

Note in particular the early account of causation in hybrid systems, using causal explanation, by Persson and Staflin [c-ecai-90-497]. Additional examples of the use of this expressiveness are found in the associated article on TAL and in example notes that are based on the present article.

The characterization of goal-directed behavior requires a few additional predicates as defined in [j-etai-1-105], [c-kr-98-304].

4 Entailment methods

Generally speaking, an entailment method is a rule that allows one to go from a collection of premises (known facts) to conclusions. Entailment methods may be expressed using inference rules, but in the CRL tradition we have usually expressed them in terms of models. Many of these entailment methods have been translated into corresponding circumscription policies in the course of the work on TAL.

4.1 Preferential entailment methods

Preferential entailment which was introduced by Bossu and Siégel [j-aij-25-13] (see also Shoham [c-ijcai-87-388]) is the simplest case of an entailment method. It is characterized by a preference relation $\ll$ on models, and maps a set Γ of premises to the set of *selected models*

$$Min(\ll, [\![\Gamma]\!])$$

where $[\![\Gamma]\!]$ is the set of classical models of Γ, and $Min(\ll, S)$ is the subset of S consisting of its $\ll$-minimal members. We write $\Gamma \mathrel{|\!\!\approx}_{\ll} \alpha$ and say that the formula α *is entailed by* Γ *according to* this entailment method iff

$$Min(\ll, [\![\Gamma]\!]) \subseteq [\![\alpha]\!]$$

Quite early it was observed that this class of entailment methods is too restrictive for practical purposes. Sandewall [c-ijcai-89-894] proposed *filter preferential entailment* as a way of dealing correctly with postdiction scenarios, where some observations refer to times other than the initial time. In filtered preferential entailment, the set of premises is divided into two *partitions* Γ_1 and Γ_2, and the set of selected models is chosen as

$$Min(\ll, [\![\Gamma_1]\!]) \cap [\![\Gamma_2]\!]$$

Concretely, Γ_2 was chosen as the set of observations pertaining to specific timepoints, and Γ_1 as the set of all other premises.

Other choices of entailment methods may be appropriate in order to obtain additional expressivity, or for better performance properties. The general formulation of entailment methods in [b-Sandewall-94], page 195 is as follows with some simplification. An *n-place* entailment method (where n is an integer ≥ 1) maps an n-tuple of sets of formulae to a set of models called the *selected models*, obtained from the n formula sets of the tuple using:

- Syntactic transformations on a set of formulae.
- The function $[\![\cdot]\!]$.
- Minimization using a preference relation on models.
- Obtaining a largest subset of a set of formulae satisfying a given property, e.g. consistency.
- Introducing additional 'system' axioms that are not part of the scenario description.
- Conventional set-theoretic operations on sets of formulae and sets of interpretations.

A recent article [j-etai-1-105] describes a four-place entailment method that is intended for characterizing goal-directed behavior in an agent. A particularly interesting entailment method is PMON [b-Sandewall-94], p. 243; [c-ecai-94-401] that also has the form

$$Min(\ll, [\![\Gamma_1]\!]) \cap [\![\Gamma_2]\!]$$

but where Γ_2 is a designated set of "system" axioms and Γ_1 contains all the premises characterizing the scenario: action instances, action laws, and observations. The initial version of PMON defined Γ_2 as the set of *nochange premises* according to the schema

$$\mathsf{C}(t, f, v) \rightarrow \mathsf{X}(t, f)$$

for all fluents f. (Several of the early papers studied the restricted case where the fluent domain was fixed, finite, and every fluent had a unique name, so that quantification over fluents was considered unnecessary. Hence the need for an axiom schema in this case). Later variants of PMON have extended the same method to more general cases, in particular PMON-R for ramification [c-ajcai-95-267] and for delayed effects [c-ecai-98-542].

Lifschitz [j-aij-74-351] formulated filtering as *nested circumscription* and showed its use for nonmonotonic applications in several areas, including for several approaches to the frame problem[1].

4.2 Non-preferential entailment methods

The above formulation of entailment methods subsumes the case where the given set of premises is augmented with additional axioms, but no preference relation at all is imposed. Explanation closure can therefore also be expressed in this framework.

[1]The reference to filtering is marginally present in Lifschitz's articles on the subject.

4.3 Entailment methods without logic

If the duration of actions is disregarded, and actions are modelled only in terms of their starting state and ending state, then each action can be understood as a function from states to sets of states, and the action law as a way of characterizing that function in logic. (Equivalently, of course, the action is understood as a binary relation on states). This view suggests that it may be possible and useful to characterize entailment methods directly on the level of discrete mathematics, and without introducing logic formulae at all. To account for a reportoire of actions, it is convenient to consider one single state-transition function $N(a, r)$ mapping an action a and a state r to a set of possible result states.

In particular, if the actual state-transition function is N, then an action law for an action with inertia corresponds to a broader function N' where $N(a, r) \subseteq N'(a, r)$, and where $r' \in N'(a, r)$ is true iff there is some $r'' \in N(a, r)$ where r' is equal to r'' in all those components (features) that are affected by the action, but r' is unconstrained otherwise.

With this definition, N' corresponds to the monotonic reading of the action laws: new values are specified for a few of the fluents; other fluents remain unspecified. The purpose of introducing nonmonotonicity is to impose the inertia assumption by reducing the set of successor states, which corresponds to reconstructing N from N'.

The article [c-kr-96-99] used this approach for formulating a number of entailment methods for ramification and for assessing their range of applicability.

4.4 Reduction of entailment methods to circumscription and to FOPC

The most important path to implementation of entailment techniques appears to be by way of circumscription. Only the TAL part of CRL have so far been concretely studied in this respect. Please see Doherty's article on TAL for the specifics.

Other implementation strategies have also been tried besides the use of circumscription. In [c-ismis-89-Sandewall] we described a tableau technique that was directly based on an entailment method for an early (three-valued) precursor of CRL.

5 Formal semantics and its use for assessments

5.1 Range of applicability

A metatheory of actions and change differs from object level theories in that the former analyses proposed entailment methods (in the context of their respective logics) with regard to their *range of applicability*, ROA. The ROA of an entailment method is defined as the set of scenario descriptions for which the method's set of selected models equals the set of *intended models* that is defined by an *underlying semantics*.

The purpose of the underlying semantics is therefore to capture our intuitions or simplifying assumptions about the matter at hand. Of course it is not possible to *prove* that the underlying semantics captures the intuitions correctly, or that the simplifying assumptions are appropriate. The range of applicability results therefore bottom out in an *informally motivated* decision as to the choice of ontological assumptions, *formally expressed.*

Each underlying semantics characterizes a certain range of expressiveness. As new expressiveness is added to the language, it becomes necessary to revise the underlying semantics, even in those cases where the base language was retained and only surface-level constructs were added.

Just as many entailment methods require the use of partitioned premise sets, so do our underlying semantics. Underlying semantics are therefore defined for scenario descriptions, which are tuples of sets of formulae as was said in subsection 2.7.

If Υ is an n-place scenario description and S an n-place entailment method, then S defines a set of *selected models* $S(\Upsilon)$ as discussed above. The underlying semantics likewise defines a set of *intended models* $Mod(\Upsilon)$. The entailment method is said to be *correct for* Υ iff

$$S(\Upsilon) = Mod(\Upsilon)$$

Consequently, the *range of applicability* for S is the set of all Υ for which the entailment method is correct.

5.2 Extending the expressiveness

As new expressiveness is imported into the language, the notion of classical models is retained unchanged, but the definition of intended models is revised. For example, concurrency and ramification each require their extensions to the underlying semantics. Notice that sometimes the syntax extension is limited to the introduction of additional abbreviations in the surface language, that is, very trivial extensions. For example, ramification and concurrency did not require any syntax extension at all, and causality with delayed effects motivated the introduction of the abbreviation $\gg$ that was defined above. However, new underlying semantics was certainly necessary.

5.3 The ontological family $\mathcal{K}$-IA

The most developed underlying semantics has been presented in [b-Sandewall-94] and characterizes systems with the following expressive capabilities and restrictions:

- Strict inertia
- Actions with duration in discrete metric time (linear or branching)
- Nondeterministic actions
- Composition of actions using sequential composition, conditionals, and repetition
- No ramification, qualification, concurrency, or delayed effects
- No surprises or external events
- The agent has complete knowledge of the current state of the world

This was a quite broad range of expressiveness compared to other approaches at the time, but of course it only covers a part of the CRL syntax. The following language domains are involved:

- The full language defined by CRL syntax

- The sublanguage of CRL covered by the mentioned semantics
- Several subsets again of that sublanguage, namely those that are the range of applicability for some particular entailment method.

The book therefore introduces a code notation for *ontological families*, that is, classes of scenario descriptions with well defined expressivity and restrictions on expressivity. The ontological family characterized here is written $\mathcal{K}$-**IA**, and subsets thereof are named systematically. In particular, the subfamily of $\mathcal{K}$-**IA** where all actions are assumed to have the duration of one time-unit is written $\mathcal{K}$-**IsA**. The subfamily where all actions are only characterized in terms of their starting state and ending state is written as $\mathcal{K}$-**IeA**. For them, there are no restrictions on the duration of the actions or the timing of state changes within the action's duration.

5.4 Underlying semantics

The full definition for $\mathcal{K}$-**IA**, which was defined in [b-Sandewall-94], requires a number of details, and here we shall only present its major aspects. We first define the semantics for the subset $\mathcal{K}$-**IsA**, and then the semantics for $\mathcal{K}$-**IA**. The notation is chosen along the lines of [c-kr-96-99].

5.4.1 Partial state-transition semantics

Let a set $\mathcal{F}$ of *features* (fluent names) be given. A *state* r is a mapping from features to corresponding values. A *history* H is a mapping from integer times (≥ 0) to states.

Let N be a state-transition function

$$actions \times states \mapsto sets\ of\ states$$

as introduced in subsection 4.3. Also, let $A = \{\langle t_i, a_i \rangle\}$ be a set of timepoint-action pairs, where $i \neq j \rightarrow t_i \neq t_j$ in order to avoid concurrency.

The set $Hist(N, A)$ of histories for N and A is defined as the set of all histories H satisfying the following conditions. For each $t \geq 0$, if $\langle t, a \rangle \in A$, then $H(t+1) \in N(a, H(t))$. Also, if no action a satisfies $\langle t, a \rangle \in A$, then $H(t+1) = H(t)$.

This means of course that the history H is formed in accordance with the state-transition function and the prescribed action occurrences, and with the default of the world remaining unchanged at the times when there is no action.

The intended models for a given scenario description $\Upsilon \in \mathcal{K}$-**IsA** can in principle be obtained as follows. First, obtain the set of classical models $\langle Holds, Occurs, Occlude \rangle$ for Υ. According to the assumption, *Occurs* will be a set of triples $\langle t, t+1, a \rangle$. Construct A as the corresponding set of pairs $\langle t, a \rangle$. For the given set of action laws, obtain the corresponding state-transition function N. The model $\langle Holds, Occurs, Occlude \rangle$ is an intended model iff there is some $H \in Hist(N, A)$ such that $Holds(t, f, v) \leftrightarrow H(t)(f) = v$, that is, the state $H(t)$ assigns the value v to the feature f, for all t, f, and v.

This tentative definition has one major weakness: it does not express the 'frame' assumption, that is, the assumption that action laws can be written so that fluents that remain unchanged need not be mentioned in the action law. In order to capture this important consideration, the definition is extended as follows. The definitions in the previous paragraphs remain valid unless changed.

Let $X(a, r)$ be a function

$$actions \times states \mapsto sets\ of\ features$$

that satisfies the following condition: if $r' \in N(a, r)$ and $f \notin X(a, r)$, then $r(f) = r'(f)$. In other words, only those features that are members of $X(a, r)$ are allowed to change their value in any of the transitions allowed by N.

For given N and X, the *partial transition function* N° is defined as a function that maps actions times states to sets of partial states, defined by

$$N^\circ(a, r) = \{r' \dagger X(a, r) \mid r' \in N(a, r)\}$$

where $r \dagger F$ is the function restriction operator. In other words, the members of the value set of $N(a, r)$ are restricted to those features that occur in $X(a, r)$. Note that the argument is still a complete state.

The purpose of this construction is to obtain an entity N° that directly corresponds to the action laws. As was discussed in section 4.3, it provides a way of escaping from the complexities of the logic formulae, to a more concise representation that in particular is syntax independent. The syntax for action laws in $\mathcal{K}$-**IsA** is defined so that the translation between them and N° is straightforward in both directions.

The function N° that is used here and the function N' defined in section 4.3 have similar motivations but differ technically. N' is used in the account of ramification. Note that X can be reconstructed from N° but not from N'.

The 'overwrite' operator $r \oplus r'$ is defined so that $(r \oplus r')(f)$ equals $r'(f)$ if the latter is defined, otherwise it equals $r(f)$.

The function *Hist* was used for the provisional definition of intended models. It is now generalized as follows for the case of partial transition functions. The set $Hist(N^\circ, A)$ of histories for N° and A is defined as the set of all histories H satisfying the following conditions. For each $t \geq 0$, if $\langle t, a\rangle \in A$, then $H(t+1) = H(t) \oplus r'$ for some $r' \in N(a, H(t))$. Also, if no action a satisfies $\langle t, a\rangle \in A$, then $H(t+1) = H(t)$.

The definition of the intended models for a given scenario description in $\mathcal{K}$-**IsA** is then finalized as follows. Obtain the set of classical models $\langle Holds, Occurs, Occlude\rangle$. *Occurs* will be a set of triples $\langle t, t+1, a\rangle$. Construct A as the corresponding set $\langle t, a\rangle$. For the given set of action laws, obtain the corresponding partial state-transition function N°. The model $\langle Holds, Occurs, Occlude\rangle$ is an intended model iff there is some $H \in Hist(N^\circ, A)$ such that $Holds(t, f, v) \leftrightarrow H(t)(f) = v$, that is, the state $H(t)$ assigns the value v to the feature f, for all t, f, and v.

5.4.2 Trajectory semantics

The partial state-transition semantics does not represent the details of how fluents may change at different points in time during the execution of an action with extended duration. It is therefore only adequate for the ontological families $\mathcal{K}$-**IsA** and (after some simple modifications) for $\mathcal{K}$-**IeA**.

The *trajectory semantics* is adequate for full $\mathcal{K}$-**IA**, and is a fairly simple generalization of the partial state-transition semantics. Whereas N° maps an action and a state to a set of partial states over a subset of the features, the trajectory transition function maps an action and a state to a set of sequences of partial states. The definitions are as follows.

A *trajectory* for a set F of features is a non-empty, finite sequence of partial states that are defined exactly over F. A *trajectory transition function* $N^\square$ for a function X, with the same structure as defined above, is a mapping from actions times states to sets of trajectories such that $N^\square(a,r)$ is a set of trajectories for $X(a,r)$.

The function *Hist* is now generalized again. Let A be a set of triples $\langle s,t,a\rangle$ where the intervals $[s,t]$ pairwise overlap in at most one point, and let $N^\square$ be a trajectory transition function. The set $Hist(N^\square, A)$ of histories for $N^\square$ and A is defined as the set of all histories H satisfying the following conditions. For each $s \geq 0$, if $\langle s,t,a\rangle \in A$ for some t and a, then there is some trajectory $w \in N^\square(a, H(s))$ that has length t-s, and where

$$H(s+i) = H(s) \oplus w(i) \textit{ for all } i \textit{ where } 1 \leq i \leq k$$

Also, for each $u \geq 0$ that does not satisfy $s \leq u < t$ for any $\langle s,t,a\rangle \in A$, it must be the case that $H(u+1) = H(u)$.

The definition of the intended models for a given scenario description Υ in $\mathcal{K}$-**IA** is finalized in complete analogy with the preceding variants. Obtain the set of classical models $\langle Holds, Occurs, Occlude\rangle$ for Υ. *Occurs* is a set of triples $\langle s,t,a\rangle$. For the given set of action laws, obtain the corresponding trajectory transition function $N^\square$. The model $\langle Holds, Occurs, Occlude\rangle$ is an intended model iff there is some $H \in Hist(N^\square, Occurs)$ such that $Holds(t,f,v) \leftrightarrow H(t)(f) = v$, that is, the state $H(t)$ assigns the value v to the feature f, for all t, f, and v.

5.5 Extensions over $\mathcal{K}$-IA

Several extensions to this underlying semantics have been made. In [c-kr-96-99], the partial state-transition semantics is extended to the case of ramification. The approach is as follows: the transition function N is represented in two parts, namely the *invocation relation* $G(a,r,r')$ and the *causal transition relation* $C(r,r')$. The full execution of an action a in a starting state r is modelled as a chain

$$G(a,r,r_1), C(r_1,r_2), C(r_2,r_3), \ldots C(r_{k\text{-}1}, r_k)$$

and if r_k has no successor according to C then $r_k \in N(a,r)$.

In [c-sss-95-Yi], Yi extended the trajectory semantics to account for concurrent actions. In [t-pisa-98-1], Brandoni extended the trajectory semantics to the case of continuous time.

Some of the syntactic constructs that are defined in the present article have not yet obtained their corresponding underlying semantics. This does not mean that they lack representational rigor, of course, but only that we do not yet have the tools for assessing the properties of their proposed entailment methods. In particular, the syntactic expressions for sensori-motoric phenomena and for agent behavior still await obtaining underlying semantics.

6 CRL Extension for Domain Objects

Most applications require a formalism that is able to refer in a systematic way to several physical objects of the same type. For example, one may wish to refer to several cars, each of which has fluents representing its current velocity, its current direction, its color, the license-plate number, who is the present driver, and so on. We use the term *domain objects* for objects in this sense, in order to distinguish them both from *description objects* such as timepoints and fluents, and from *data objects* that only exist computationally.

In conventional logic, such a requirement is met by going from propositional to predicate logic, and the ramifications of that step are well understood. In CRL, like in any current logic for reasoning about actions and change, the extension is trivial in one sense and nontrivial in another.

With respect to syntax and classical semantics, the matter is quite straightforward. All it takes is to introduce one or more additional domains for domain objects, and to allow functions that map domain objects or combinations of them, to fluent names. It is natural to allow domain objects to be first-class citizens, which means there must be object constant symbols, object variable symbols, and so forth. These are the aspects that we chose to exclude at the beginning of this article. The DFL logic in the book "Features and Fluents" was defined so as to allow a single class of domain objects. The carefully worked-out definition of TAL in [f-linep-97-20] contains a complete account of the required definitions.

As one defines the underlying semantics and the nonmonotonic aspects of domain objects, two phenomena are worth mentioning:

- Many practical scenarios are characterized by a finite and relatively small number of domain objects. This may be e.g. the set of cars or the set of persons that participate at once in a given scene. The scenario therefore has a closure property with respect to the objects: one is able to enumerate and name the objects in the scene; only those objects that are mentioned in this way need to be assumed to exist for the purpose of the argument.
- General statements about the application, such as action effect laws and domain constraints, apply across all scenarios and independently of the number of domain objects in them. For example, the laws about the effects of a *move* action in a blocks world apply regardless of the number of blocks in the scenario at hand.

The first observation shows that such scenarios are inherently only propositional: it is possible to replace a finite formalization using explicit domain objects by a larger but still finite formalization (except for the use of an infinite time domain) that does not use domain objects. For example, the fluents $colorof(car1)$ and $colorof(car2)$ may be replaced by the "atomic" fluents $Colorof\_Car\_1$ and $Colorof\_Car\_2$, provided that every proposition containing a universal quantification over objects is replaced by a corresponding conjunction of instances, one for each of the finitely many objects, and similarly for existential quantifications. In that sense, the "propositional" case of these logics is applicable to a larger class of practical examples than what might strike the eye at first.

The second observation modifies the picture somewhat. As long as we are satisfied with only reasoning about one single scenario at a time, we can fix the set of domain objects and treat the formalization as essentially propositional. However, if instead we wish to reason about *all* scenarios, or at least *several* scenarios with a varying number of domain objects, it is no longer possible to use the closed-world assumption on domain objects. For example, if we should wish to use logic for deriving consequences, in blocks worlds, of using *move* and other similar actions, and if we should wish our conclusions to hold regardless of the number of domain objects, we do have a "first-order" situation. Then it is not possible to fix the domain, and the problem is no longer essentially propositional.

7 Summary

Cognitive Robotics Logic (CRL) is an extensible logic language for characterizing actions and change, in particular for use in cognitive robotics. Its development emphasizes the issues of syntax, expressivity, underlying semantics and entailment methods (defined in terms of the semantics). Development of proof methods is de-emphasized. The salient results from this approach refer to the range of applicability and other related properties of the entailment methods. These results constitute a metatheory of actions and change.

CRL is syntactically defined as a base language and a surface language. The base language is characterized by the following aspects:

- its three major predicates *Holds*, *Occurs*, and *Occlude*, which provide coherence when the language is extended;
- its reportoire of categorial functions, which is augmented when additional expressiveness is required in the language.

The surface language provides additional notational convenience, and is defined by translation to the base language.

The range of expressivity includes actions with duration, nondeterministic actions, actions in hybrid worlds with piecewise continuous fluents, some forms of ramification and causation, imprecise sensors and actuators, action failure, and some aspects of goal-directed agent behavior.

Entailment methods are functions that map scenario descriptions to sets of *intended models*. They are defined using a reportoire of set-theoretic operations on sets of formulas and sets of models, including but not restricted to minimizing a set of models with respect to a preference relation.

A progression of underlying semantics is defined, beginning with the partial state-transition semantics and its immediate generalization, the trajectory semantics. These underlying semantics are used for the formal analysis of the range of applicability of various entailment methods, including both those proposed by the others in this research, and those that developed in the course of the present work.

References:

[b-Sandewall-94] Erik Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems.* Oxford University Press, 1994.

[c-ajcai-95-267] Patrick Doherty and Pavlos Peppas. *A comparison between two approaches to ramification: PMON(R) and $\mathcal{AR}_0$.* Proc. Australian Joint Conference on Artificial Intelligence, 1995 (X. Yao, ed.), pp. 267-274.

[c-ecai-90-497] Tommy Persson and Lennart Staflin. *A Causation Theory for a Logic of Continuous Change.* Proc. European Conference on Artificial Intelligence, 1990 (Luigia Carlucci Aiello, ed.), pp. 497-502. Pitman Publishing.

[c-ecai-94-401] Patrick Doherty. *Reasoning about Action and Change using Occlusion.* Proc. European Conference on Artificial Intelligence, 1994 (Anthony G. Cohn, ed.), pp. 401-405. John Wiley ¡amp¿ Sons.

[c-ecai-98-542] Lars Karlsson, Joakim Gustafsson, and Patrick Doherty.

Delayed Effects of Actions. Proc. European Conference on Artificial Intelligence, 1998, pp. 542-546.

[c-fcs-98-198] John McCarthy. *Elaboration tolerance.* Proc. Formalization of Commonsense Reasoning, 1998, pp. 198-216.

[c-hart-97-3] Erik Sandewall. *Relating high-level and low-level action descriptions in a logic of actions and change.* Proc. Hybrid and Real-Time Systems, 1997 (Oded Maler, ed.), pp. 3-17. Springer Verlag.

[c-ijcai-87-388] Yoav Shoham. *Nonmonotonic Logics: Meaning and Utility.* Proc. International Joint Conference on Artificial Intelligence, 1987, pp. 388-393. Morgan Kaufmann Publishers, Inc..

[c-ijcai-89-894] Erik Sandewall. *Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds.* Proc. International Joint Conference on Artificial Intelligence, 1989, pp. 894-899. Morgan Kaufmann Publishers, Inc..

[c-ismis-89-Sandewall] Erik Sandewall. *A decision procedure for a theory of actions and plans.* Proc. International Symposium on Methodologies for Intelligent Systems, 1989. North-Holland.

[c-ismis-93-558] Erik Sandewall. *Systematic assessment of temporal reasoning methods for use in autonomous agents.* Proc. International Symposium on Methodologies for Intelligent Systems, 1993, pp. 558-570. North-Holland.

[c-kr-89-412] Erik Sandewall. *Combining Logic and Differential Equations for Describing Real-World Systems.* Proc. International Conf on Knowledge Representation and Reasoning, 1989, pp. 412-420. Morgan Kaufmann Publishers, Inc..

[c-kr-96-99] Erik Sandewall. *Assessment of ramification methods that use static domain constraints.* Proc. International Conf on Knowledge Representation and Reasoning, 1996, pp. 99-110. Morgan Kaufmann Publishers, Inc..

[c-kr-98-304] Erik Sandewall. *Logic Based Modelling of Goal-Directed Behaviou.* Proc. International Conf on Knowledge Representation and Reasoning, 1998, pp. 304-315. Morgan Kaufmann Publishers, Inc..

[c-sss-95-Yi] Choong-Ho Yi. *Towards the Assessment of Logics for Concurrent Actions.* Proc. Stanford Spring Symposium, 1995. AAAI Press.

[f-linep-97-14] Lars Karlsson and Joakim Gustafsson. *Reasoning about actions in a multi-agent environment.* Linköping University Electronic Press, Series Computer and Information Science, 1997, Nr. 14.

[f-linep-97-20] Patrick Doherty. *PMON+: A Fluent Logic for Action and Change: Formal Specification, Version 1.0.* Linköping University Electronic Press, Series Computer and Information Science, 1997, Nr. 20.

[f-linep-98-15] Patrick Doherty, Joakim Gustafsson, Lars Karlsson, and Jonas Kvarnström. *TAL: Temporal Action Logics Language Specification and Tutorial.* Linköping University Electronic Press, Series Computer and Information Science, 1998, Nr. 15.

[j-aicom-9-214] Erik Sandewall. *Towards the validation of high-level action descriptions from their low-level definitions.* AI Communications, vol. 9

(1996), pp. 214-224.

[j-aij-25-13] Geneviève Bossu and Pierre Siégel. *Saturation, Nonmonotonic Reasoning, and the Closed World Assumption.* Artificial Intelligence Journal, vol. 25 (1985), pp. 13-65.

[j-aij-64-337] Ray Reiter. *Proving properties of states in the situation calculus.* Artificial Intelligence Journal, vol. 64, pp. 337-351.

[j-aij-74-351] Vladimir Lifschitz. *Nested abnormality theories.* Artificial Intelligence Journal, vol. 74 (1995), pp. 351-365.

[j-aij-77-249] Murray Shanahan. *A circumscriptive calculus of events.* Artificial Intelligence Journal, vol. 77 (1995), pp. 249-284.

[j-etai-1-105] Erik Sandewall. *Logic-Based Modelling of Goal-Directed Behavior.* Electronic Transactions on Artificial Intelligence, vol. 1, pp. 105-128.

[j-jlp-17-301] Michael Gelfond and Vladimir Lifschitz. *Representing action and change by logic programs.* Journal of Logic Programming, vol. 17 (1993), pp. 301-321.

[s-Gabbay-94-439] Erik Sandewall and Yoav Shoham. *Nonmonotonic Temporal Reasoning.* In: Dov M. Gabbay, C.J. Hogger, and J.A. Robinson (ed): Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 4: Epistemic and Temporal Reasoning, pages 439-498. Oxford University Press.

[t-pisa-98-1] Sergio Brandano. *A Logic-Based Calculus of Fluents.* Technical report, Dipartimento di Informatica, Università di Pisa (Italy), 1998, Nr. 1.