LiTH-IDA-R-88-08 ISSN-0281-4250

# INSTITUTIONEN FÖR DATAVETENSKAP
## UNIVERSITETET OCH TEKNISKA HÖGSKOLAN
## LINKÖPING

# Formal Semantics for Reasoning about Change with Ramified Causal Minimization

by

Erik Sandewall



RESEARCH REPORT
RKLLAB, April 1988

*Postadress:*
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

*Mailing address:*
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

# Computer and Information Science Department
# Linköping University

## PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

( *Linköping Studies in Science and Technology. Dissertations.* )

*No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982.

*No 94 **Hans Lunell:** Code Generator Writing Systems, 1983.

*No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983.

*No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984.

*No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons. 1987.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning. 1987.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems. 1987.

## Licentiate of engineering Theses:

(*Linköping Studies in Science and Technology. Theses.*)

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.

No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems. 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1985.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System. 1987.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs. 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behaviour of the Greedy Triangulation. 1986.

No 104 **Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology. 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-based Editing. 1987

No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments. 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge. 1987.

No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs. 1987

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems. 1987

LiTH-IDA-R-88-08 ISSN-0281-4250

# Formal Semantics for Reasoning about Change with Ramified Causal Minimization

by

Erik Sandewall
email: ejs@ida.liu.se or ejs@liuida.uucp

Abstract: We deal with the frame problems, including the qualification problem, by requiring that for every state change there is a corresponding action causing it, and minimize the set of actions. An action may cause other actions occuring at the same time, or afterwards. The syntax and semantics are specified for a logic where this approach to the frame problem can be expressed stringently, and the proper minimization criterium is formulated.

One characteristic feature of this work is that sub-structures which are used in the semantics, such as partial states and descriptors, are also "first class citizens" from the syntactic point of view. In this way we obtain a kind of reification, but with less formal overhead than when formulas are reified directly.

*This research was supported by the Swedish Board of Technical Development.*

*Postadress:*
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

*Mailing address:*
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

# 1. Introduction.

Reasoning about time, action, and change is important for several purposes within A.I., for example for knowledge based planning (= "planning" in the A.I. sense of the word) and for natural language understanding. It is widely recognized that reasoning about change requires non-monotonic reasoning devices for dealing with the frame problems. (We here and henceforth use the term "frame problems" in a broad sense, including also the ramification problem etc). Furthermore it is now recognized that it is far from trivial to render reasoning about change in terms of available non-monotonic reasoning systems (Hanks and McDermott, 1986, 1987). Finally we recognize that the semantics of non-monotonic reasoning systems can be uniformly treated in terms of preference ordering on models (Shoham, 1986B), sets of models, or other model theoretic entities.

The appropriate approach to reasoning about change is therefore a model theoretic one: first define what interpretations we wish to use and the rules for evaluating formulas in such interpretations; then define a preference ordering on such interpretations; and only then (if ever) define an inference system which obtains those conclusions which are entailed in maximally preferred models. Shoham (1986B) used this strategy for his choice of reasoning principle, namely chronological ignorance.

In this paper we apply the same strategy to another reasoning principle namely ramified causal minimization. Haugh (1987) and independently Lifschitz (1987) have argued very convincingly that simple causal minimization is preferable to chronological minimization for the prototypical examples. Causal minimization is based on the following observation "in the intended model, all changes in the values of fluents are caused by actions" (Lifschitz, 1987, p. 967).

In general, one must of course minimize, rather than rule out unsupported state changes. This may be done either directly or indirectly. The direct method is to minimize a relation that is strongly tied to every state change, but relative to axioms which force the relation to be true in the supported cases. The indirect method is to require *every* state change to have an action which causes it, but to minimize over actions, and to have a reportoire of "default explanations" such as "something went wrong with the battery" which will appear in the minimal action set when we are faced with observed or assumed, otherwise unexplainable state changes.

We prefer the indirect method and minimize the set of actions, particularly in order to extend the use of causal minimization for handling the ramification problem. Consider the following classical example: the cup is on the saucer, the saucer is on the table, and therefore the cup is above the table. Now the saucer is moved to the next table. How should the reasoning system realize that the cup is still on the saucer, but no longer above the first table?

The conventional approach is to have axioms or other 'rules' which link the action of moving the saucer, not only to its immediate pre- and postconditions (in this case obtaining the conclusions about the saucer's own new position), but which also link the same action to its indirect effects, in this case that the cup is above another table. In addition, the general rule that the cup can not be above two different tables (which are beside each other), is used to defeat the default of the cup staying above the old table.

However, what if the saucer is moved very rapidly, so that the cup falls off the saucer, and stays on (and therefore, above) the first table? We believe that such alternative developments (of which there are abundantly many) are best handled by assuming *several* actions. The action of moving the saucer is linked to its direct consequences, but also to other, consequent actions. Thus, if the saucer is moved carefully enough, it 'causes' another action, namely "the cup goes with the saucer" which in its turn has the effect that the cup comes to be above table 2, but no longer above table 1. Other ways of moving the saucer cause other actions for the cup, for example "the cup falls off the saucer", which update the current state in their own ways.

We use the term *ramified causal minimization* for the principle of minimizing the set of actions, and requiring an action causing every state change, in order to deal with the frame problems including the ramification problem. The identification of preferred models may be quite complex, particularly if the occurrence of an action depends on aspects of the current state.

# 2. Choice of logical approach.

The use of ramified causal minimization requires a logic which allows parallel actions, i.e. not being restricted to sequential execution of actions. This rules out the direct use of traditional advise-taker-based systems which are strictly sequential.

Shoham's semantics does allow for parallel actions. It is developed in two stages. First he develops a semantics for reified temporal logic encoded in a first order theory (section 2.2 in Shoham 1986B, also published separately as 1986A). This system, which we shall refer to as RTL, centers around the use of temporal propositions. In RTL, a *temporal proposition* is a wff of the form

TRUE(t1, t2, R(a1,a2,...))

and is used to express that the *proposition type* R(a1,a2,...) holds from time t1 to time t2. Its truth value is determined as

$\langle a1^*, a2^*, ...\rangle \in M5(t1^*, t2^*, R)$

where a1*, a2*,... are the object domain members obtained as values of the terms a1, a2, etc; t1* and t2* are the time points obtained as values of the temporal terms t1 and t2, and M5 maps

W x W x R

to relations, W being the domain of time points and R being the set of "relation symbols". W, R, and M5 are of course among the components of the interpretation.

In RTL one can express both time-dependent properties, such as "the wall is red at time t1", and actions such as "John paints the wall green from time t2 to time t3". However the RTL semantics does not require any relationship between time- dependent properties and actions. For example, it is perfectly possible that all time-dependent properties develop in the same way from time t2' to t3' as from t2 to t3 (i.e. the history from t2 to t3 repeats itself identically from t2' to t3') and yet the following two wff may have different truth value:

TRUE(t2, t3, R(a1,a2,...))

and

TRUE(t2', t3', R(a1,a2,...))

As a result RTL has great flexibility but lacks specificity. Shoham demonstrates how proposition types can be classified with respect to their interval inheritance characteristics, i.e. whether the same proposition type also holds for sub- intervals or super-intervals of the first given interval. In

particular what we here call actions would be temporal propositions formed from one particular case of Shoham's proposition types, viz. the ones he calls solid. However for the purpose of specifying ramified causal minimization, we wish to focus on exactly those proposition types, and attach the extra information about preconditions and postconditions to them. Proposition types other than the solid ones we consider as being of less interest. Also, since we are going to minimize the set of actions (corresponding to temporal propositions), we are in general not enthusiastic about mechanisms which generate a lot of spurious occurrences of them.

Reichgelt (1987) has introduced a semantics for reified temporal logic which he claims is an improvement over RTL with respect to expressiveness. We have not considered using his approach since it seems by far too complex, mostly due to the reification. The semantics proposed here is much simpler than his.

Later chapters of (Shoham 1986B) introduce an extension to RTL called CI, a nonmonotonic logic of temporal knowledge. It introduces a modal operator $\Box$ with the essential effect that propositions may be not only true or false, but also unknown. The preference ordering on interpretation is defined so that chronologically maximally ignorant models are preferred.

For modelling ramified causal minimization we need to make other kinds of additions. In order to represent preconditions and postconditions on the semantic level, we need partial states as one of the semantic domains. Furthermore partial states are used for modelling values while an action is going on: if John goes from L1 to L2 during the time period from t1 to t2, then his position at intermediate time-points is assigned the value "unknown" in the interpretation. Such considerations have led us away from RTL or other previously proposed systems, and to the system which will now be presented.

# 3. Syntax.

We subscribe to the principle that one shall develop ontology and semantics first, and then select a matching syntax. The present work has also been done in that order. However for presentation purposes it is better to describe the syntax first.

We use first-order formulas where the terms may be of the following syntactic types:

temporal terms, t
state terms, s
action descriptor terms, A
action terms, a
object descriptor terms, B
object terms, b
propositional descriptor terms, C
propositional terms, c

State terms refer to a state of the world, allowing not only total but also partial states. Propositional terms have the "truth"-value T, F, or U (for Unknown). If c is a propositional term then c=T is a wff but c itself is not. Descriptors are syntactically similar to fluents, but have a different semantics. The value of an object descriptor term is always the same object descriptor; the value of an object descriptor in a state is an object or the quasi-object u (for unknown); the value of an object term is always the same object.

By way of example, the rule "if the object b moves from l1 to l2 then its position is l1 at the beginning of the move, and l2 at the end of the move" is written as:

$$\text{atyp}(a) = \text{Move}(b,l1,l2) \rightarrow \text{val}(\text{Pos}(b),\text{bc}(a)) = l1 \wedge \text{val}(\text{Pos}(b),\text{ec}(a)) = l2$$

which would read as "if the action occurrence a is a move action with the parameters b, l1, l2, then the value of the object descriptor Pos(b) in the beginning-state (beginning-condition) of a is l1, and the value of the same object descriptor in the ending-state of a is l2". The term Move(b,l1,l2) refers to an action descriptor, which is used like Shoham's propositional types. If b is moved from l1 to l2 repeatedly, then there are several action occurrences with the same action descriptor.

In general, we have the following operators:

| | |
|---|---|
| $r(t) = s$ | state of the world at time point t |
| $val(B,s) = b$ | value of the object descriptor B in state s |
| $val(C,s) = c$ | value of the propositional descriptor C in state s |
| $bt(a) = t$ | beginning time of the action occurrence a |
| $et(a) = t$ | ending time of the action occurrence a |
| $bc(a) = s$ | precondition of the action occurrence a |
| $ec(a) = s$ | postcondition of the action occurrence a |
| $fc(a) = s$ | prevail condition of the action occurrence a |
| $atyp(a) = A$ | action type of the action occurrence a |

as well as the following relations besides equality:

| | |
|---|---|
| $t \leq t'$, $t < t'$ | temporal order on time-points |
| $s \sqsubseteq s'$ | specificity order on states (s contains less information than s' or is equal) |

The following axioms are obviously desired:

$$bt(a) < et(a)$$
$$bc(a) \sqsubseteq r(bt(a))$$
$$ec(a) \sqsubseteq r(et(a))$$
$$s \sqsubseteq s' \rightarrow (val(B,s) = val(B,s') \vee val(B,s) = u)$$

For another example, consider the action of toggling a lamp, i.e. to turn it on if it is off, and turning it off if it is on. The challenge in this example is to express action effects which also depend on the present state of the world; it was discussed in (Lifschitz 1987). In our notation it is characterized by

$$atyp(a) = Toggle(b) \rightarrow val(Ison(b),bc(a)) = neg(val(Ison(b),ec(a)))$$

where Toggle(b) is an action descriptor, Ison(b) is a propositional descriptor whose value is T in a state if the lamp b is turned on there, and neg(T) = F, neg(F) = T, neg(U) = U.

In general, then, we assume that formulas are written using the operators which have just been described, plus application specific operators of the kinds that will be defined in the next section.

Clearly this syntax is sufficiently expressive for characterizing actions including those that proceed in parallel. It is also able to express general statements about time and action, of the type "effects can not precede their causes", which Reichgelt quote as a major reason for his semantics.

# 4. Semantics.

We deal with partial interpretations, where total interpretations are a special case. An interpretation is a tuple

$\langle \langle T, D, \ldots \rangle, \langle tp, Q, R \rangle \rangle$

The first part of the interpretation characterizes the domain as follows:

T set of time-points

D (domain) set of objects

and additional elements will soon be added. We let u, the unknown object, be a distinct object not in D, and define

$D^+ = D \cup \{u\}$

The second part of the interpretation characterizes a specific history in that domain. tp characterizes the ordering on the time points as a mapping from T x T to either of the following eight ordering indicators: $<$, $=$, $>$, $\leq$, $\neq$, $\geq$, unknown, contradiction. (These eight indicators are lattice-ordered as in figure 1, which is used for defining a partial order $\sqsubseteq$ on time point orderings tp)

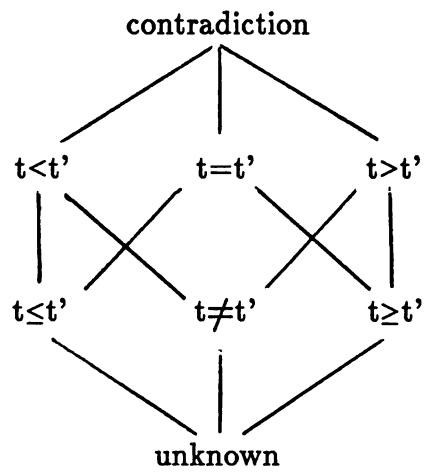


Figure 1.

The element Q in the interpretation is a plan, i.e. a set of actions. R is a process i.e. a mapping from time-points to states.

A state is in principle the same thing as an interpretation in first order logic. However we bring the definition of states in line with the interpretation of descriptors by the following two, trivial changes from current practice:

*Fluents vs. descriptors*: a fluent is usually seen as a mapping from time-points to objects or truth-values (McCarthy and Hayes, 1969). This means that if John and Jim are twins born on the same day, then the fluents

Age(john)

and

Age(jim)

are the same fluent, since they are the same sets of argument- value pairs. We instead define them as descriptors so that

Age(john) = ⟨"Age", john⟩

whereby Age(john) $\neq$ Age(jim) iff john $\neq$ jim. A descriptor thus is a tuple where the first element is a functor, and subsequent elements are members of D.

Notice that a descriptor is not just a reified formula, since its non-first elements are objects in the domain rather than terms.

*States:* an interpretation in first-order logic is usually seen as a mapping from functors (function symbols, relation symbols) to functions or relations of the appropriate types. We equivalently consider a state as a mapping from tuples of the form

⟨functor, object, object,... ⟩

to either D+ or {T,F,U}. In this way, a state is simply a mapping from descriptors as just described, to their values, allowing also partial knowledge.

We adopt the convention that capitalized function symbols are descriptor valued, and always result in a tuple where the function symbol itself is the first element, and the evaluated arguments are the following elements. This rule applies to function symbols such as Age, Move, Pos, Toggle, Ison in the examples above.

We are now ready to define the full list of elements of an interpretation as

⟨⟨T,D,AG,BG,CG,BF,CF,si,G⟩, ⟨tp,Q,R⟩⟩

where now

| | |
|---|---|
| T | is a set of time-points |
| D | is a set of objects |
| AG | is a set of functors forming action descriptions, e.g. Move |

| | |
|---|---|
| BG | is a set of functors forming object descriptions, e.g. Age |
| CG | is a set of functors forming propositional descriptions, e.g. Ison |
| BF | is a set of rigid function symbols, e.g. fatherof(b) |
| CF | is a set of rigid predicate symbols, e.g. isbrother(b,b') |
| si | is a state used as the static interpretation, which provides definitions for the members of BF and CF. For example if d2 and d3 are members of D we could have si(⟨isbrother,d2,d3⟩) = T |
| G | is a relation which expresses the constraints on actions such as preconditions, postconditions, etc. It will be introduced in the next section. |

Similarly a *dynamic state* is a state which provides definitions for the members of BG and CG. In the second part of the interpretation, which we call the history, the *process* R is a mapping from T to dynamic states.

The *plan* Q, finally, is a set of action occurrences; each action occurrence being a tuple

⟨t1, ⟨h, f, b, e⟩, t2⟩

where t1 and t2 are members of T indicating when the action starts and ends; h is an action descriptor formed in the same way as the other kinds of descriptors for example

⟨Move,b,l1,l2⟩

and f, b, and e are dynamic states. If a is this action occurrence, we write

bt(a) = t1
et(a) = t2
bc(a) = b
ec(a) = e
fc(a) = f
atyp(a) = h

The definition of the syntax of logical formulas can now be completed: formulas are written using the operators defined in the previous section, and the members of the sets AG, BG, CG, BF, CF specified here.

The rules for obtaining the truth value of a wff in an interpretation are now largely obvious. They become particularly simple due to the ways descriptors were defined and used. The various syntactic types evaluate into the following domains:

temporal terms: T

state terms: dynamic states
action descriptor terms: descriptors where the first element is in AG
action terms: Q (i.e. the set of action occurrences)
object descriptor terms: descriptors formed from BG
object terms: D+
propositional descriptor terms: descriptors formed from CG
propositional terms: {T,F,U}
(We assume that the distinction between the domain T and the truth-value T is always clear by context)

Terms formed using capitalized functors evaluate into tuples as just described

Terms formed using members of BF and CF are object terms and propositional terms, respectively, and are evaluated using si in the obvious way

Terms formed using the operators bt, et, bc, ec, fc, and atyp are evaluated as just defined

Terms formed using the operator r are evaluated using the R component of the interpretation

Terms of the form val(B,s) are evaluated as

$$s^*(B^*)$$

where s* is the value of s, and B* is the value of B. Terms of the form val(C,s) are evaluated in the same way.

Formulas using the binary relations $<$, $\leq$, etc. are evaluated using the temporal order specification tp in the obvious way.

Formulas using the information content ordering $\sqsubseteq$ are evaluated using the obvious flat lattices on objects and truth- values, and the derived lattice on states, in the same way as in e.g. Sandewall and Rönnquist, 1986.

# 5. Consistency constraints on histories.

We now proceed to introduce those constraints which relate actions to their preconditions, postconditions, etc., and which we shall call the constraint of plan-consistency. For this purpose we introduce the following concepts:

An action-instantiation is a triple ⟨f,b,e⟩ of dynamic states.

The component G of the interpretation is a binary relation on action-descriptors x action-instantiations. In the toggle example above we would have

G(Toggle(b), ⟨empty, on, off⟩)
G(Toggle(b), ⟨empty, off, on⟩)

where on is the dynamic state where

on(Ison(b)) = T

and all other descriptors are mapped to U or u; off is the dynamic state where

off(Ison(b)) = F

and all other descriptors are mapped to U or u; and empty is the dynamic state which maps all descriptors to U or u.

The following consistency requirements, called *plan-consistency*, are now imposed on interpretations. For every action occurrence

⟨t1, ⟨h,f,b,e⟩, t2⟩

in Q we require:

1. that G(h, ⟨f,b,e⟩), i.e. that the action occurrence is in the reportoire of available action occurrences defined by G;

2. that $t1 < t2$ according to tp;

3. that the preconditions, postconditions, and prevail conditions are respected, i.e.

$b \sqsubseteq r(t1)$
$e \sqsubseteq r(t2)$
for each t such that $t1 \leq t \leq t2$, $f \sqsubseteq R(t)$

These constraints on interpretations are easily expressed using conventional axioms (conventional in the sense of not using e.g. non-monotonic logic).

# 6. The preference order of ramified causal minimization.

We can now proceed to the characterization of ramified causal minimization itself. As the first step, simple causal minimization imposes the following constraint: if t < t' and

$$val(B,r(t)) \neq val(B,r(t'))$$

then there must be some action which defines the change of value through its precondition/postcondition requirements.

A plan-consistent interpretation which also satisfies this condition is called *plan-complete* (since all value changes are accounted for by the actions).

The extension from simple to ramified causal minimization imposes a preference ordering on interpretations: if two interpretations i and i' contain the plans Q and Q', and

$$Q \subset Q'$$

then i is preferable over i'.

The criterium for selecting preferred models is therefore to select those (or that) interpretations which are maximally preferable, among those plan-complete interpretations which also satisfy the domain axioms.

This concludes the semantic specifications. From here the natural next step would be to identify a suitable, presumably non-monotonic reasoning system that matches the spec. In doing so, one may of course transform the semantics in various ways. For example if one prefers to minimize over a predicate rather than a domain then it is trivial to convert the process p from a set of tuples into a relation. For example if Q is seen as a relation

$$q(t1, h, j, t2)$$

where j is an action-instantiation, one would require

$$q(t1, h, j, t2) \rightarrow G(h,j)$$

and minimize q.

# 7. Discussion.

One of the features of the logical system proposed here, is that a number of semantical constructs are made available also to the syntactical level. This applies for descriptors and for states which are formed from descriptors: they play key roles in the semantics but they also are "first class citizens" from the syntactic language point of view. Indeed it is this very feature that allows us an expressiveness which otherwise is often only achieved using reification.

Yet there are also significant differences of perspective between the syntactic and semantic levels, particularly with respect to the action occurrences. On the syntactic level, an action is characterized through its action descriptor, for example Toggle(b). Presumably the action descriptor will correspond closely to what we think of as one type of actions. On the semantic level, the relation G relates the action descriptor to the various instantiations which characterize how the action may be carried out. However, the set of instantiations is not simply the relation between states used e.g. in Shoham's M5 function (as described above), or as used by Georgeff (1986). Instead each instantiation expresses one of the major cases, such as (in the example) "the lamp was on and is turned off" and "the lamp is off and is turned on". Only those conditions which are relevant for an action are specified in the instantiation; other are mapped to the unknown object or truth- value by the three states in the instantiation. This is how the conditions and the effects of the action are identified in the semantics.

For another example, consider the action of painting the box red (regardless of its previous color). The obvious way of characterizing that action on the syntactic level is

$$\text{atyp}(a) = \text{Paint}(d,\text{red}) \rightarrow \text{val}(\text{Color}(d),\text{ec}(a)) = \text{red}$$

where we may of course also replace the rigid constant symbol red by a variable throughout. On the semantic level it would at first seem natural to associate Paint(d,red) with an instantiation $\langle f,b,e \rangle$ where

$$\text{val}(\text{Color}(d),f) = \text{val}(\text{Color}(d),b) = u$$
$$\text{val}(\text{Color}(d),e) = \text{red}$$

However when the properties of these models are analyzed (as in Sandewall and Rönnquist, 1986), it is in fact better to associate Paint(d,red) with a set of more specific instantiations $\langle f,b,e \rangle$ where in one of them

val(Color(d),b) = green

in another one it is blue, and so on for all the available colors. The syntactic level characterization of the action is of course valid in both the first instantiation and in the later, more specific ones.

The technical reason for using the more specific instantiations is that they facilitate the definitions of how actions "update" the current state. If the precondition b and the postcondition e assign values to the same set of descriptors, then effects of an action-instantation on a current state which subsumes the preconditions, is obtained by first "subtracting" the preconditions, and then "adding" the postconditions.

Such technical reasons are however not the only reasons for identifying the different instantiations of an action descriptor. When an autonomous system perceives an ongoing action, or when it is to execute it, or when the consequences of an action must be analyzed, we believe that it is useful to deal work with the particular instantiation of the action.

# 8. Properties of plan-complete interpretations.

Suppose for a moment that the ordering of the time-points is total, and that the plan Q and the state R(t0) for an initial time-point t0 ($\leq$ the starting time of all actions in Q) are given. Under a number of fairly natural assumptions, there is then at most one plan-complete interpretation which contains that plan Q, and whose process R assigns the same state to R(t0). The process R is obtained by, in a certain sense, "executing" the plan Q with R(t0) as the initial state, subtracting preconditions to the current state, adding postconditions, and checking that prevail conditions apply continuously.

Proceed then to the more general case where the ordering of the time-points is partial and not total. Is it then still possible to make a meaningful definition of R(t) in subsequent time-points? Problems may occur e.g. if the plan contains parallell actions which change the value of the same "property" or proposition.

In (Sandewall and Rönnquist, 1986), we have identified static conditions on the plan which guarantee that no such problems will occur. Those conditions essentially guarantee that actions which "update the same propositions", i.e. whose preconditions and postconditions affect the same descriptors, must necessarily appear in sequence.

# 9. Model-based reasoning.

Another result has also emerged from this work. Traditionally, one has considered the semantics only as a formal device which is used to define the meaning of formulas, and to verify the soundness and completeness of the inference system. It has likewise been assumed that computation should be carried out in terms of syntactic and proof-theoretic devices, i.e. the inference system. Current catchwords like "inference engine" only confirm the traditional view. We have found, however, that the interpretations defined in this paper, which also allow partial states and partial temporal orderings, can in themselves be subject to useful computation. The various requirements on interpretations, such as plan-consistency and plan-completeness, can be directly interpreted using a constraint system. In this way we obtain a computational device which is stringently defined, and which is able to perform reasoning about time and action without the use of an inference engine.

It is not clear at this point how such model-based reasoning compares to the more traditional inference methods, in terms of expressiveness and performance. It is easy to construct examples where constraint propagation in partial models comes very naturally, and the use of an inference system seems like a detour. It is likewise possible to construct examples where the model based reasoning becomes very clumsy, for example because we obtain artificially many instantiations. We do not yet know the trade-off for practical cases.

## References.

Georgeff, Michael P.: Actions, Processes, and Causality. In: M. Georgeff and A. Lansky (eds): Reasoning about Action and Plans, pp. 99-122. Morgan-Kaufmann, 1986.

Hanks, Steve, and McDermott, Drew: Default reasoning, non- monotonic logics, and the frame problem. Proc. AAAI, 1986, pp. 328-333.

Hanks, Steve, and McDermott, Drew: Nonmonotonic logic and temporal projection. Artificial Intellience, Vol. 33, Nr. 3, pp. 379-412, Nov. 1987.

Haugh, Brian A.: Simple Causal Minimizations for Temporal Persistence and Projection. Proc. AAAI, 1987, pp. 218-223.

Kautz, H. The logic of persistence. Proc. IJCAI 1985, pp. 401- 405.
Lifschitz, Vladimir: Formal Theories of Action (preliminary report). In F.Brown (ed): The Frame Problem in Artificial Intelligence. Morgan Kaufmann, 1987.

McCarthy, John, and Hayes, Patrick: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Machine Intelligence, Vol. 4, pp. 463-502, 1969.

Poole, D.: On the comparison of theories: preferring the most specific explanation. Proc. IJCAI 1985.

Reichgelt, Hans: Semantics of a reified temporal logic. In: J. Hallam and C. Mellish (eds): Advances in Artificial Intelligence, pp. 49-61. Proceedings of AISB 1987 Conference. Wiley and Sons, 1987.

Sandewall, Erik, and Rönnquist, Ralph: A representation of Action Structures, Proc. AAAI 1986.

Shoham, Yoav (1986A): Reified Temporal Logics: Semantical and Ontological Considerations. Proc. ECAI 1986, pp. 390-397.

Shoham, Yoav (1986B): Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence. Dissertation, Yale University, 1986.

Avdelning, institution, fakultet
Division, department, faculty

Department of Computer and
Information Science
Linköping University

ISBN:

ISSN: ISSN-0281-4250

Rapportnr:
Report no: LiTH-IDA-R-88-08

Upplagans storlek:
Number of copies:

Datum:
Date: April 1988

Projekt:
Project:

Titel:
Title: Formal Semantics for Reasoning about Change with Ramified Causal Minimization

Författare:
Author Erik Sandewall

Uppdragsgivare:
Commissioned by:

Dnr.:
Call no:

Rapporttyp:
Kind of report:

- ☐ Examensarbete/Final project
- ☐ Delrapport/Progress report
- ☐ Reserapport/Travel report
- ☐ Slutrapport/Final report
- ☒ Övrig rapport/Other kind of report

Rapportspråk:
Language:

- ☐ Svenska/Swedish
- ☒ Engelska/English
- ☐ _____

Sammanfattning (högst 150 ord):
Abstract (150 words):

Abstract: We deal with the frame problems, including the qualification problem, by requiring that for every state change there is a corresponding action causing it, and minimize the set of actions. An action may cause other actions occuring at the same time, or afterwards. The syntax and semantics are specified for a logic where this approach to the frame problem can be expressed stringently, and the proper minimization criterium is formulated.

One characteristic feature of this work is that sub-structures which are used in the semantics, such as partial states and descriptors, are also "first class citizens" from the syntactic point of view. In this way we obtain a kind of reification, but with less formal overhead than when formulas are reified directly.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

# A Selection of Previous Research Reports.

LiTH-IDA-R-87-26 **Jonas Löwgren:** Applying a Rapid Prototyping System to Control Panel Dialgues.

LiTH-IDA-R-87-24 **Sven Moen:** Drawing Dynamic Trees.

LiTH-IDA-R-87-22 **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues. To appear in *Proc. of 14th Conference of the Association for Literary and Linguistic Computing*, Göteborg 1-5 June, 1987.

LiTH-IDA-R-87-21 **Harold W. Lawson, Jr.:** Challenges and Directions in Computers and Education.

LiTH-IDA-R-87-20 **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications. This paper is to appear in *Microprocessing and Microprogramming, the Euromicro Journal*, 1988.

LiTH-IDA-R-87-19 **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications. Also in *Proc. of the 10th World Computer Congress*, Dublin, Ireland, Sept. 1-5, 1986.

LiTH-IDA-R-87-18 **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-Based Systems.

LiTH-IDA-R-87-17 **Tony Larsson:** Specification and Verification of VLSI Systems Actional Behaviour This is a close version of a paper presented at *the 8th international conference on Computer Hardware Description Languages, CHDL*, 87.

LiTH-IDA-R-87-16 **Shamsul I. Chowdhury:** State of the Art in Statistical Expert Systems. Also in *Proc. of the Conference on Expert Systems and their Applications*, Avignon, May 1987.

LiTH-IDA-R-87-15 **Nils Dahlbäck:** Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?

LiTH-IDA-R-87-14 **Peter Fritzson:** Window System Architectures - an Overview. Also presented at *A Tutorial on Software Development Environments*, Linköping University, Nov 24-25 1986.

LiTH-IDA-R-87-13 **Mats Wirén:** A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. Also in *Proc. of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, April 1—3, 1987.

LiTH-IDA-R-87-11 **Sture Hägglund, Christer Hansson, Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures. Also in *Proc. of the 3rd Int. Conf. on Expert Systems*, London, 1987.

LiTH-IDA-R-87-10 **Andrzej Lingas:** On Parallel Complexity of the Subgraph Isomorphism Problem.

LiTH-IDA-R-87-09 **Andrzej Lingas, Marek Karpinski:** Subtree Isomorphism and Bipartite Perfect Matching are Mutually NC Reducible.

LiTH-IDA-R-87-08 **Christos Levcopoulos, Andrzej Lingas, Jörg-R. Sack:** Nearly Optimal Heuristics for Binary Search Trees with Geometric Generalizations. Also in *Proc of ICALP'87*, Karlsruhe, July 1987.

LiTH-IDA-R-87-07 **Peter Haneclou:** A Formal Approach to Reason-maintenance Based on Abstract Domains.

LiTH-IDA-R-87-06 **Johan Hultman:** COPPS - A Software System for Defining and Controlling Actions in a Mechanical System.

LiTH-IDA-R-87-05 **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembly Processes.

LiTH-IDA-R-87-03 **Mikael R.K. Patel:** A Threaded Interpretive Language Supporting Programming in the Large. Also in *Proc. of The sixth Rochester Forth Conference*, University of Rochester, Rochester, New York, June 11-14, 1986.

LiTH-IDA-R-87-02 **Rober Bilos:** A Token-Based Syntax Sensitive Editor. Also presented at the *Workshop on Programming Environments - Programming Paradigms*, Roskilde, Denmark, October 22-24, 1986.

# The Department of Computer and Information Science
# Linköping University

organizes undergraduate and graduate studies in *Computer Science*, *Telecommunication and Computer Systems*, and *Administrative Data Processing*. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- *ACTLAB - Laboratory for Complexity of Algorithms*, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.
- *AIELAB - Artificial Intelligence Environments Laboratory,* which conducts research on representation and manipulation of knowledge, problem solving techniques and expert systems with mechanical engineering applications.
- *ASLAB - Application Systems Laboratory,* which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.
- *CADLAB - Laboratory for Computer-Aided Design of Electronics*, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.
- *LIBLAB - Laboratory for Library and Information Science,* which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.
- *LOGPRO - Laboratory for Logic Programming*, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.
- *NLPLAB - Natural Language Processing Laboratory*, which conducts research related to the development and use of natural language interfaces to computer software.
- *PELAB - Programming Environments Laboratory*, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.
- *RKLLAB - Laboratory for Representation of Knowledge in Logic*, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

## Research Reports 1988

| | |
|---|---|
| LiTH-IDA-R-88-08 | **Erik Sandewall:** Formal Semantics for Reasoning about Change with Ramified Causal Minimization. |
| LiTH-IDA-R-88-07 | **Erik Sandewall et all:** IDA Annual Report 1987. |
| LiTH-IDA-R-88-06 | **Christer Bäckström:** A Representation of Coordinated Actions Characterized by Interval Valued Conditions. |
| LiTH-IDA-R-88-05 | **Christer Bäckström:** Keeping and Forcing: How to Represent Cooperating Actions. |
| LiTH-IDA-R-88-04 | **Wlodek Drabent, Simin Nadjm-Tehrani, Jan Maluszynski:** Algorithmic Debugging with Assertions. |
| LiTH-IDA-R-88-03 | **Staffan Bonnier, Jan Maluszynski:** Towards a Clean Amalgamation of Logic Programs with External Procedures. |
| LiTH-IDA-R-88-02 | **Ulf Nilsson:** Inferring Restricted AND-Parallelism in Logic Programs using Reduction. |
| LiTH-IDA-R-88-01 | **Jan Maluszynski:** Logic Programs Introducing S-Unification. A Processing Letters, North |