LiTH-IDA-R-88-28

ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP
UNIVERSITETET OCH TEKNISKA HÖGSKOLAN
LINKÖPING

Non-Monotonic Entailment for Reasoning about Time and Action Part II: Concurrent Actions

by

Erik Sandewall

RESEARCH REPORT
RKLLAB, September 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

The Department of Computer and Information Science
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
No 139	**Christer Bäckström:** Reasoning about Interdependent Actions. 1988.
No 140	**Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing. 1988.

LiTH-IDA-R-88-28 ISSN-0281-4250

Non-Monotonic Entailment for Reasoning about Time and Action Part II: Concurrent Actions

by

Erik Sandewall
E-mail: ejs@ida.liu.se

Completed while visiting at the Computer Science Department,
Stanford University

Abstract. An explicit temporal logic for reasoning about time and ac- tion is extended in order to account for concurrency between actions. The extensions are such that in no model can several actions co-occur if they control the same property (i.e. influence it so that it changes or may change). Also, several actions which require the same property to be held at the same value are allowed to co-occur, but they can not in turn co-occur with even one action that controls that property. In this way, traditional concepts in the study of concurrent processes have been imported into the formal-logical framework.

This research was supported by the Swedish Board for Technical Development, Rank Xerox AB, and the Rockwell Corporation.

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 and Topic.

The previous paper in the present trilogy introduced an explicit temporal logic for non-monotonic reasoning about time and action, and demonstrated the viability of the logic for several kinds of temporal reasoning. An explicit temporal logic is one where explicit reference to time-points is possible, so that the following are examples of logical formulas (wffs):

$$Holds(t_1, load, t_2)$$

$$t_2 \prec t_3$$

$$Holds(t_3, fire, t_4)$$

One of the advantages of such a logic is that it is able to express synchronuous occurrence of actions. However, as soon as we begin using that property of the logic, we run into the classical concurrency issues. For example, what conclusions should we obtain, and expect to obtain, from the following axiom:

$$Holds(t_1, load, t_2) \wedge Holds(t_1, fire, t_2)$$

combined with axioms specifying the effects of actions of the types *load* and *fire*. These questions were left open in the previous paper, and are the topic of the present paper. We show a method of incorporating the classical concepts[1] of exclusive and non-exclusive access to a resource, into the temporal logic. This is accomplished by introducing two auxiliary relations, *Controls* corresponding to excluive access, and *Steadies* corresponding to non-exclusive access, together with the appropriate axioms. A prerequisite of the approach is that actions must be syntactically reified in the logic.

The general problem is as follows: if we have formulated the conditions and effects for action types under the assumption that no other actions which affect the same properties (called *conflicting actions*) occur at the same time, what conclusions should we obtain if some other such action does occur synchronously.

In the context of A.I. approaches to temporal reasoning, it may be argued that this problem is related to the qualification problem, since the co-occurrence of conflicting actions could be an example of the kinds of exceptions that the qualification problem is concerned with. This however assumes that co-occurrence is an exceptional phenomenon, and clearly in some environments it is not. In the present paper we therefore prefer a more direct approach.

2. Controlled properties: concurrent actions competing to change a property.

In the approach used here, we stipulate that two conflicting actions *can not* occur at the same time, and write axioms which make sure that that

[1] see e.g. the survey paper by Andrew and Schneider [AS83]

does not happen in a model. The philosophical rationale is that if e.g. the holder of the gun performs the movements associated with loading and firing at the same time, then we consider that there is only an instance of a third type of action, $load + fire$, which is going on, and not a *load* action, not a $fire$ action. The effects of $load + fire$ must then be specified separately (for example from an understanding of the mechanical properties of the gun).

If $load + fire$ is not included in the reportoire H of action types, then the result will be, for planning purposes, that we can not construct plans where one attempts to *load* and $fire$ at the same time, since such a plan would be inconsistent. For situation understanding purposes (diagnostics, story understanding), the logic system can not even represent the concept of attempting to do them together, except as a contradictory statement. This may be a crude way of solving the problem, but everything that is more sophisticated is also going to be much harder.

Since models have been defined to be those interpretations which satisfy all axioms in $\Gamma_s \cup \Gamma_a \cup \Gamma_t$, the appropriate way to express this condition on models is by writing it as an axiom (or axiom schema), to be added to the set Γ_t of general axioms. However, in order to be able to express that condition, we must first reify actions, i.e. introduce actions as an additional syntactic type.

To this end, a new relation $Controls(a, c)$ is introduced and is inserted as an intermediate step in the derivation of *Notpersists* conclusions. Whereas in part I the axioms that specify effects of actions were written on the form

$$Holds(t, h, u) \rightarrow \ \ldots \ Notpersists(t, c, u) \ \ldots$$

we will now write them as

$$Holds(a, t, h, u) \rightarrow \ \ldots \ Controls(a, c) \ \ldots$$

and add the following axiom schema

$$Controls(a, c) \wedge Holds(a, t, h, u) \rightarrow Notpersists(t, c, u).$$

which we shall call $\Gamma_{t,2}$, to the set Γ_t of general temporal axioms. This in itself does not change what *Notpersists* conclusions can be drawn, but it allows us to also introduce the temporal ordering constraint as another axiom schema $\Gamma_{t,3}$:

$$Controls(a, c) \wedge Controls(a', c) \rightarrow$$
$$(Holds(a, t, h, u) \wedge Holds(a', t', h', u') \rightarrow a = a' \vee u \preceq t' \vee u' \preceq t).$$

Thus the only purpose of the changes is to be able to express the concept of conflicting actions formally, and to introduce constraints against their co-occurrence. This deals with the problem when two different, concurrent actions "want to" change the value of the same property.

There is also a problem when an action "wants" a property to stay constant, and therefore can not concur with other actions which "want to" change it, but it is able to concur with other axioms which also "want"

the property to stay constant, and with the same value. We shall return to that problem in the next section, but before that we specify in an incremental fashion what are the necessary changes to the syntax and semantics of the logic, based on what has been said here.

MODIFICATIONS TO THE LOGIC.

We have seen that the relation *Holds* for action arguments is extended with one more argument a, which is a variable for the new type of actions. Thus if we write

$$Holds(a_2, t_3, dig, t_6)$$

we are saying that the *action type* "dig" has had a particular instance in the *action* a_2. The instance lasted from t_3 to t_6. The action type *dig* of course does not have a beginning point or an end point since it potentially refers to many instances at different times.

The following formal changes are made:

a. In the syntax, a set AC of action constant symbols and a set AV of action variable symbols are introduced, disjoint from each other and from the other elementary syntactic domains.

b. Among logical formulas, the following additional kinds are allowed:

$$Holds(a, t, h, u).$$

$$Controls(a, c).$$

where $a \in AC \cup AV$.

c. In interpretations, the component X is supplemented by a component

$$Y \subseteq P \times C$$

which is used to interpret *Controls* in the obvious way.

d. Also in interpretations, a component $M_A : AC \mapsto P$ which interprets AC is introduced. Variable assignments are extended so that they assign time-points to members of TV, and members of P to members of AV.

e. Formulas $Holds(a, t, h, u)$ are interpreted to be T iff

$$val(a) = \langle val(t), h, val(u) \rangle$$

which then is a member of P. $Controls(a, c)$ is T iff

$$\langle val(a), c \rangle \in Y$$

f. Action specification axioms are re-written so that they imply $Controls(a, c)$ for every property c that the action controls, for example:

$$Holds(a, t, fire, u) \rightarrow$$
$$Notholds(t, gunloaded, u) \vee$$
$$[Holds(t, gunloaded) \wedge Notholds(t, fredalive, u) \wedge$$
$$\neg Holds(u, gunloaded) \wedge Controls(a, gunloaded)] \vee$$

$$[Holds(t, gunloaded) \wedge Holds(t, fredalive) \wedge$$
$$\neg Holds(u, gunloaded) \wedge \neg Holds(u, fredalive) \wedge$$
$$Controls(a, gunloaded) \wedge Controls(a, fredalive)].$$

g. The axiom schema $\Gamma_{t,2}$ defined above is added to the set Γ_t of axioms for temporal reasoning in general.

h. The definition of $<$ is modified so that we primarily minimize Y ($Controls$), secondarily maximize X ($Persists$), tertiarily minimize $abch(J)$, and fourthly minimize the set P.

It is easily verified that the logic we obtain after these modifications obtains the same results under preferential entailment as what we had before. Also and as intended, after these changes we can express the constraint against conflicting actions by adding axiom schema $\Gamma_{t,3}$ to the set Γ_t of general axioms for temporal reasoning.

3. Steadied properties: properties that one or more actions require to be held constant.

There is also another and related problem which comes up when actions are performed in parallel. What conclusions should we draw from the following set of axioms:

$$Holds(t_1, landplane, t_3).$$

$$Holds(t_2, leaveplane, t_4).$$

$$t_1 \prec t_2.$$

$$t_2 \prec t_3.$$

with the obvious meanings and specifications of these action types: *landplane* means that the aeroplane lands, requires that it is airborne at the beginning of the action and not airborne at its end, and controls the property *airborne*. The action type *leaveplane* can only occur when the plane is not airborne, and changes the property *onboard* (referring to pilots and/or passengers being in the plane) from true to false.

With these action specifications, the given scene axioms are certainly satisfiable, namely in interpretations where the plane lands so quickly that it is never airborne after t_2. Also if the action axioms are written with the same pattern as in earlier sections, the *Controls* relation and its axioms do not introduce any inconsistency: *landplane* must control *airborne* since it changes it, but *leaveplane* does not need to control *airborne*.

For some applications, such as for story understanding or scene understanding, one may be quite prepared to accept these axioms and interpretations. However for other applications and particularly for planning there is a problem since the planner or plan execution system may not have total

control of how the *landplane* action is performed. The action may be performed by a manual, mechanical or electronic sub-system which does it at its own pace, and so the plane may still be airborne when passengers start leaving. Since this is obviously an issue at least for some important types of application, we address now the question of how to deal with it in the present framework.

A possible solution is simply to let *leaveplane* also control *airborne*, which forces the plan to be made in such a way that $t_3 \preceq t_2$. Unfortunately, that also disallows having several parallel actions (such as *unload-baggage* or *connect-vacuum*), all of which rely on prevailing non-airborne-ness, but none of which changes the plane's flight status.

Therefore we introduce instead one more relation $Steadies(a, c)$ which is similar to *Controls* except that it is "sharable": a property c can be *steadied* by several concurrent actions but it can not be both *controlled* by one action and *steadied* by another at the same time. One can think of "steadies" as analoguous to non-exclusive ("read") access in software systems, and "controls" as exclusive ("read and write") access.

At this point we do not address the question whether the action requires the property to be steady, or causes it to be steady. A calculus which makes that distinction explicitly has been described by Bäckström [Bä88].

With this new device the axioms for *landplane* and *leaveplane* would be as follows:

$$Holds(a, t, landplane, u) \rightarrow$$
$$[Controls(a, airborne) \wedge Holds(t, airborne) \wedge$$
$$\neg Holds(u, airborne)].$$

$$Holds(a, t, leaveplane, u) \rightarrow$$
$$[Steadies(a, airborne) \wedge Notholds(t, airborne, u) \wedge$$
$$Controls(a, onboard) \wedge Holds(t, onboard) \wedge$$
$$\neg Holds(u, onboard)].$$

Notice that the formula $Steadies(a, airborne)$ is used to express an assumption that the value of the property *airborne* is kept constant at either T or F, not necessarily at T. In the present case it is F throughout the steady period.

With this change, interpretations must be extended by one more relation Z which is formed in the same way as Y and used in order to interpret *Steadies* in the obvious way, and which is minimized together with Y. The axiom schema $\Gamma_{t,3}$ for *Controls* which was introduced above is now strengthened to the following:

$$Controls(a, c) \wedge (Controls(a', c) \vee Steadies(a', c)) \rightarrow$$

$$(Holds(a,t,h,u) \wedge Holds(a',t',h',u') \rightarrow$$
$$a = a' \vee u \preceq t' \vee u' \preceq t).$$

These are all the axioms that are needed for temporal reasoning in specific examples, using semantic entailment. There are of course also formulas which are T in all interpretations by virtue of how the semantics has been defined, for example

$$Holds(a,t,h,u) \wedge Holds(a,t',h',u') \rightarrow t = t' \wedge h = h' \wedge u = u'.$$

Such formulas would be of interest if we define an inference mechanism and look for an axiomatization, but that is outside the scope of the present work.

4. Alternative approaches.

The approach that we have described now amounts to little more than importing, into the logic, some classical concepts from the theory of concurrent processes, namely the concepts of exclusive and non-exclusive access. Let us consider whether one could achieve the same purpose with less machinery, and in particular without introducing the constraint supporting relations *Controls* and *Steadies*.

A possible alternative to the approach just described would be to *define* the ending-time of actions in such a way that they end immediately when the changes caused by the action have taken effect. For example the action *loadgun* would extend over successive time-points where the gun is unloaded, and end at the first time-point when the gun is loaded. In that way the impossibility of loading and firing during overlapping intervals would follow from the properties of the actions themselves, without the need for auxiliary relations such as *Controls*.

That approach however breaks down as soon as one has an action which controls several properties, which do not all change their value at the same time. For example, the action of loading the gun also has the effect of reducing the number of cartridges in the supply box, but that property changes strictly before the gun becomes operational.

Another possible alternative would be to define actions operationally so that they[2] wait until all preconditions are satisfied before they start to operate. In the aeroplane landing example, the leaveplane *action* would start at time t_2, but the leaveplane *procedure* would remain dormant until time t_3 when the plane has landed. That approach is quite natural from the perspective of programming languages, but very unnatural for common-sense reasoning. A knowledge-based planning system should produce a plan for when actions take place, not for when they should start waiting.

[2] i.e. the device or procedure performing the action

These difficulties of alternative approaches add to the reasons for importing the traditional synchronization concepts from the theory of concurrent programming, into the temporal logic.

5. Revised semantics and axioms.

Let us now summarize the revised logic after the extensions defined in the last two sections. Comparing to the definitions in part I and part II, we have essentially only made extensions rather than direct changes.

SYNTAX

Let the following be given:
C, a set of properties;
H, a set of action types;
TC, a set of time-point constant symbols;
TV, a set of variable symbols for time-points;
AC, a set of action constant symbols;
AV, a set of variable symbols for actions.

The set of logical formulas is defined inductively to consist of the following:

$$t = u$$

$$t \prec u$$

$$Holds(a, t, h, u)$$

$$Holds(t, c, u)$$

$$Persists(t, c, u)$$

$$Notpersists(t, c, u)$$

$$Controls(a, c)$$

$$Steadies(a, c)$$

$$\alpha \wedge \beta$$

$$\neg\alpha$$

$$\forall v \alpha$$

and, for notational convenience also the ones listed under "extended syntax" below, as well as formulas using the ordinary reportoire of propositional connectives. The syntactic variables in the definitions are supposed to range as follows: $t, u \in TC \cup TV$; $a \in AC \cup AV$; $h \in H$; $c \in C$; α, β are logical formulas; $v \in TV \cup AV$.

SEMANTICS

An *interpretation* is a tuple $\langle \mathbf{T}, tp, R, P, M_T, M_A, X, Y, Z \rangle$, where

$\mathbf{T}$ is a nonempty, enumberable set of timepoints

$tp \subseteq \mathbf{T} \times \mathbf{T}$ is a strict total order

$R \subseteq \mathbf{T} \times C$

$P \subseteq \mathbf{T} \times H \times \mathbf{T}$

M_T is a mapping $TC \mapsto \mathbf{T}$

M_A is a mapping $AC \mapsto P$

$X \subseteq \mathbf{T} \times C$

$Y, Z \subseteq P \times C$

We assume that $\mathbf{T}$ and tp behave like integers, as before. A *variable assignment* VA is the union of a mapping $TV \mapsto \mathbf{T}$ and a mapping $AV \mapsto P$.

The truth-value $J \models \alpha_{[VA]}$ of a formula α in an interpretation J under the variable assignment VA, is T under the following conditions, and F otherwise:

If α has the form $t = u$: iff $val(t) = val(u)$

If α has the form $t \prec u$: iff $\langle val(t), val(u) \rangle \in tp$

If α has the form $Holds(a, t, h, u)$: iff $val(a) = \langle val(t), h, val(u) \rangle \in P$

If α has the form $Holds(t, c, u)$: iff $t \preceq t' \preceq u \rightarrow \langle val(t'), c \rangle \in R$

If α has the form $Persists(t, c, u)$: iff $t \preceq t' \prec u \rightarrow \langle val(t'), c \rangle \in X$

If α has the form $Notpersists(t, c, u)$: iff $t \preceq t' \prec u \rightarrow \langle val(t'), c \rangle \notin X$

If α has the form $Controls(a, c)$: iff $\langle val(a), c \rangle \in Y$

If α has the form $Steadies(a, c)$: iff $\langle val(a), c \rangle \in Z$

Here $val(t)$ and $val(a)$ refers to the value of a temporal term and an action term, determined in the obvious way using M_T, M_A, and VA. Propositional connectives and quantifiers are interpreted as usual.

EXTENDED SYNTAX.

The following formulas are also admitted by the syntax, and are defined to evaluate like the corresponding, "expanded" formulas:

$t \preceq u$ has same value as $t = u \vee t \prec u$

$Holds(t, h, u)$ has same value as $\exists a.Holds(a, t, h, u)$

$Holds(t, c)$ has same value as $Holds(t, c, t)$

$Notholds(t, c, u)$ has same value as $t \preceq t' \preceq u \rightarrow \neg Holds(t', c)$

AXIOMS.

In each application we use the union of three sets of axioms, as follows.

Γ_s, the *scene axioms*, shall be ground formulas (i.e. containing no variables) formed using the temporal relations ($\prec$, $\preceq$, $=$) or the relations *Holds* or *Notholds*, and (when occasionally needed) disjunction of such formulas.

Γ_a, the *action specification axioms*, shall be formulas of the form

$$Holds(a, t, h, u) \rightarrow \alpha_1 \vee \alpha_2 \vee ...$$

with one or more disjuncts on the consequent side. The formula is said to be a *conventional* action specification iff each disjunct α_i has the form

$$\beta_{i,1} \wedge \beta_{i,2} \wedge ...$$

where again each conjunct $\beta_{i,k}$ has either of the following forms:

$$Controls(a, c) \wedge \pm Holds(t, c) \wedge \pm Holds(u, c).$$

$$Steadies(a, c) \wedge Holds(t, c, u).$$

$$Steadies(a, c) \wedge Notholds(t, c, u).$$

where $\pm Holds$ stands for either *Holds* or $\neg Holds$, and either of the two $\pm Holds$ literals may be omitted. The a, t, and u must be the same as are used in the antecedent of the implication. All action specifications in the examples we have had, are conventional.

Γ_t, the *general temporal axioms*, is the set of all instances of the following axiom schemas:

$$Holds(a, t, h, u) \rightarrow t \prec u.$$

$$Controls(a, c) \wedge Holds(a, t, h, u) \rightarrow Notpersists(t, c, u).$$

$$Controls(a, c) \wedge (Controls(a', c) \vee Steadies(a', c)) \rightarrow$$
$$(Holds(a, t, h, u) \wedge Holds(a', t', h', u') \rightarrow$$
$$a = a' \vee u \preceq t' \vee u' \preceq t).$$

For future reference, these are called $\Gamma_{t,1}$, $\Gamma_{t,2}$, and $\Gamma_{t,3}$, respectively.

PREFERENCE ORDERING ON INTERPRETATIONS.

Let

$$J_1 = \langle \mathbf{T}, tp, R_1, P_1, M_{T,1}, M_{A,1}, X_1, Y_1, Z_1 \rangle$$

$$J_2 = \langle \mathbf{T}, tp, R_2, P_2, M_{T,2}, M_{A,2}, X_2, Y_2, Z_2 \rangle$$

be two interpretations. We define the two preference orderings as follows:

$$J_1 \ll J_2 \leftrightarrow R_1 = R_2 \wedge P_1 = P_2 \wedge$$
$$((Y_1 \subset Y_2 \wedge Z_1 \subseteq Z_2) \vee (Y_1 \subseteq Y_2 \wedge Z_1 \subset Z_2) \vee$$
$$(Y_1 = Y_2 \wedge Z_1 = Z_2 \wedge X_1 \supset X_2))$$

$$J_1 < J_2 \leftrightarrow [abch(J_1) \subset abch(J_2)] \vee$$

$$[abch(J_1) = abch(J_2) \wedge P_1 \subset P_2]$$

where the set of abnormal changes, $abch(J)$, is defined like before as

$$\{\langle t, c\rangle \mid \langle t, c\rangle \in X \wedge [\langle t, c\rangle \in R \leftrightarrow \langle succ(t), c\rangle \notin R]\}$$

These preference orders are then used for preferential entailment $\models_{\ll,<}$ just like before.

An example of the use of these techniques is prepared as a separate paper.

References

[AS83] Gregory R. Andrews and Fred B. Schneider. Concepts and notations for concurrent programming. *Computing Surveys*, 15(1), March 1983.

[Bä88] Christer Bäckström. Reasoning about interdependent actions. Master's thesis, Computer Science Department, Linköping University, 1988.

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-88-28

Upplagans storlek:
Number of copies:

Datum:
Date: September 1988

Projekt:
Project:

Titel:
Title: Non-Monotonic Entailment for Reasoning about Time and Action
Part II: Concurrent Actions

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. An explicit temporal logic for reasoning about time and action is extended in order to account for concurrency between actions. The extensions are such that in no model can several actions co-occur if they control the same property (i.e. influence it so that it changes or may change). Also, several actions which require the same property to be held at the same value are allowed to co-occur, but they can not in turn co-occur with even one action that controls that property. In this way, traditional concepts in the study of concurrent processes have been imported into the formal-logical framework.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

A Selection of Previous Research Reports.

LiTH-IDA-R-88-19 **Lin Padgham, Ralph Rönnquist:** From a Technical to a Humane Environment: A Software System Supporting Cooperative Work. Also in *Proc. of the GDI Internationa Conference on USER INTERFACES,* Rüschlikon, Switzerland, October 20-21, 1986.

LiTH-IDA-R-88-18 **Nils Dahlbäck:** Mental Models and Text Understanding - a Commented Review.

LiTH-IDA-R-88-17 **Bengt Lennartsson:** Multi-Level Software Architectures - Industrial Experiences and Research Directions. This report is a revised version of a paper in *Advance Working Papers of The Second International Workshop on Computer-Aided Software Engineering.* Cambridge, Massachusetts. July 12-15, 1988.

LiTH-IDA-R-88-16 **Lin Padgham:** A Model and Representation for Type Information and Its Use in Reasoning with Defaults. Also in *Proc. of AAAI'88, American Association for Artificial Intelligence*, 1988.

LiTH-IDA-R-88-15 **Lin Padgham, Ralph Rönnquist:** LINCKS: An Imperative Object Oriented System. Also in *Proc. of the 20th Annual Hawaii Int. Conf. on System Sciences*, Hawaii, 1987.

LiTH-IDA-R-88-13 **Erik Tengvald:** Ett kartorienterat planeringssystem för autonoma farkoster, en design diskussion.

LiTH-IDA-R-88-12 **Erik Sandewall, Johan Hultman, Erik Tengvald:** Software Architecture and Programming Paradigms for Robotics Applications.

LiTH-IDA-R-88-11 **Mats Wirén:** An Incremental Chart Parser for PATR.

LiTH-IDA-R-88-10 **Mats Wirén:** A Control-Strategy-Independent Parser for PATR. Revised version of paper appearing in *Proc. of First Scandinavian Conf. on Artificial Intelligence*, Tromsö, Norway, March 9 - 11, 1988.

LiTH-IDA-R-88-09 **Peter Fritzson:** Incremental Symbol Processing.

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 al:** 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. Also in *Proc. of META 88 - Workshop on Meta Programming*, Bristol, June 21-24 1988.

LiTH-IDA-R-88-03 **Staffan Bonnier, Jan Maluszynski:** Towards a Clean Amalgamation of Logic Programs with External Procedures. Accepted to the *Joint Fifth Int. Logic Programming Conf. and Fifth Logic Programming Symp.*, Seattle Washington, August 15-19, 1988.

LiTH-IDA-R-88-02 **Ulf Nilsson:** Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.

LiTH-IDA-R-88-01 **Jacek Leszczylowski, Staffan Bonnier, Jan Maluszynski:** Logic Programming with External Procedures: Introducing S-Unification. A shorter version is to appear in *Information Processing Letters*, North Holland.

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. Also 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.

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.
- *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-88-28 **Erik Sandewall:** Non-Monotonic Entailment for Reasoning about Time and Action. Part II: Concurrent Actions.

LiTH-IDA-R-88-27 **Erik Sandewall:** Non-Monotonic Entailment for Reasoning about Time and Action. Part I: Sequential Actions.

LiTH-IDA-R-88-26 **Lin Padgham:** NODE: A Database for Use by Intelligen Systems. Also in *Proc. of ISMIS '88,* Torino Italy, October 12-15, 1988.

LiTH-IDA-R-88-23 **Johan Fagerström, Yngve Larsson:** Two Contributions on Debugging Distributed Systems. Also in *Proc. of ACM SIGPLAN/SIGOPS, Workshop on Parallel and Distributed Debugging*, Madison, Wisconsin, May 5-6, 1988.

LiTH-IDA-R-88-22 **Johan Fagerström:** Design and Test of Distributed Applications. Also in *Proc. of the International Conference on Software Engineering, ICSE*, Singapore, April 1988.

LiTH-IDA-R-88-21 **Johan Fagerström, Lars Strömberg:** A Paradigm for Distributed System Design and Test. Also in *Proc. of COMPCON'SPRING 88*, San Francisco, 1988.