LiTH-IDA-R-88-29 ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP

UNIVERSITETET OCH TEKNISKA HÖGSKOLAN LINKÖPING

Non-Monotonic Entailment for Reasoning about Time and Action Part III: Decision Procedure

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-88-29 ISSN-0281-4250

Non-Monotonic Entailment for Reasoning about Time and Action Part III: Decision Procedure

by

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

Completed while visiting at the Computer Science Department,
Stanford University

Abstract. A semantics-based decision procedure for reasoning about time and action is described. Given a temporal logic where$\Gamma \models_{\ll,<} \alpha$ is defined using preference semantics, the decision procedure identifies a set of partial interpretations (or interpretation schemas) which are $\ll$, $<$-preferred models for all axioms in Γ. Conclusions α can then be tested by evaluating them in the computed partial models. The paper also describes how safe control rules (i.e. rules which do not change the result of the computation, but which eliminate unnecessary branches from the search) are entailed by the general (not scene-specific) axioms, using a special variant of preferential entailment.

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 Summary.

This paper proposes a model-theoretically based decision procedure for non-monotonic reasoning about actions and their effects. The procedure uses constraint propagation in partial interpretations. Being model-theoretic, it does not require inference rules or a complete axiomatization.

The major computational approach to non-monotonic reasoning has previously been through reason maintenance systems [Doy79,dK86]. These systems have however generally been designed for propositional, non-temporal logic, and are not capable of reasoning about time and action in a systematic way. Also the formal semantic bases for reason maintenance systems are generally obscure. The method described here appears capable of handling problems of non trivial size, and is strictly defined and verified.

We base the method on the logic for temporal reasoning described in the previous papers, part I and II, of the present trilogy. Those papers define interpretations for temporal logic in such a way that they contain in essence, (1) a strict total order over a domain of time-points, (2) for each pair of time-point and "property" (e.g. "gun is loaded"), either of the truth-values T or F, (3) a set of actions, each specified as a triple $\langle t, h, u\rangle$, where t and u are time-points, and h is an action type (e.g. "load the gun"), and (4) a set of "expectations for no change", which also is a relation on time-points $\times$ properties. In addition the interpretations contain devices for controlling concurrency.

Two preference orders $\ll$ and $<$ are defined over such interpretations, where $\ll$ maximizes expectations that properties (propositional fluents) will not change, and $<$ minimizes the set of points where they change contrary to expectation, and the set of actions.

The method can be understood as follows. Consider the domain of pairs $\langle P, \xi\rangle$, where P is a set of actions which is one part of the interpretations, and ξ is a function expressing expectations for no change. The secondary preference ordering $<$ defines an order on such pairs, so that a pair is smaller (more preferred) if it contains stronger no-change assumptions, or if it contains a smaller set P. The given scene axioms define the minimal possible set(s) P, namely consisting of actions which are explicitly stated to occur.

The algorithm now searches the domain of such pairs, starting with the minimal member(s). In each node it determines whether the given axioms are satisfiable, i.e. whether there is some interpretation J which contains the given P, and for which $abch(J)$ has the same (or higher) expectations for no-change than ξ. If there exist some such J, then the $\ll$-minimal interpretations for the current P and ξ are the entailed ones, and all interpretations in $<$-larger nodes are not preferentially entailed. If on the other hand there does not exist any such J in the present node, then all $<$-successor nodes must be considered instead.

For the types of axioms we defined in parts I and II of these papers, the

problem is inherently finite when P is held fixed. Therefore the problem of determining satisfiability for given P and ξ can be done using semantic tableaux. Also we shall see that the minimization with respect to the $\ll$ ordering can be easily integrated with the semantic tableaux technique.

The processing of building the semantic tableau is seen here as the process of extending a partial interpretation to successively $\sqsubset$-larger partial interpretations, where $\sqsubset$ is the ordering on partial interpretations specifying "more content". A key part of the method is therefore to define partial interpretations in such a way that the temporal ordering between times can be partial, and need not be a total order as in parts I and II of the paper. Our definition of partial interpretations makes it possible to represent the contents of the scene axioms as one or a few partial interpretations. The other axioms are then used for gradually adding more information to the initial p.i:s. Thus the method can also be understood as a kind of constraint propagation method.

One important advantage of this method is that if the search is done in the right way, it reasons very naturally for concrete problems. Therefore, as an introduction to the later and more formal sections, section 2 of this paper contains a protocol of detailed common-sense reasoning about a simple problem. In the last sections of the paper we will refer to aspects of this protocol in order to motivate various traits of the algorithm. In particular, section 5 describes a method for obtaining "control" rules, i.e. rules which do not change the result of the computation, but whichdirects it in such a way as to eliminate the need to search certain branches in the tableaux which are unnecessary in view of the $\ll$ preference.

2. An example.

Let us first go through a concrete example, namely the perennial Yale turkey shoot. We shall repeat the axioms from previous parts of the paper, and show how a preferred partial model can be constructed, using the axioms for constraint propagation.

The set Γ_s of scene descriptions are as follows:

$Holds(t_1, load, t_2).$

$t_2 \prec t_3.$

$Holds(t_3, fire, t_4).$

$\neg Holds(t_1, gunloaded).$

$Holds(t_1, fredalive).$

The set Γ_a of action specifications consists of the following two conventional axioms, henceforth referred to as Γ_{load} and Γ_{fire}, written with the concurrency control literals from part II:

$Holds(a, t, load, u) \rightarrow$

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

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

The set Γ_t of general temporal reasoning axioms were specified in part II of the paper, and consists of instances of the axiom schemas $\Gamma_{t,1}$, $\Gamma_{t,2}$ and $\Gamma_{t,3}$.

The purpose of the exercise is to verify that these axioms together preferentially entail the surprising fact that

$$\neg Holds(t_4, fredalive).$$

and more importantly, to do the verification with a method which generalizes to a reasonable range of axioms and which is systematic enough to be implementable.

To this end, we use a working structure which captures what conclusions have been drawn so far, and which we call a *time and action map,* or *t-a-map.* We will later recognize it as a partial interpretation. Informally however, the action map looks as in the example in figure 1, which shows the action map at the end of the verification process for the example. It consists of four parts. The topmost part, labelled **T**, identifies the time-points which are being considered, in this case t_1 through t_4. The second part, labelled *tp* identifies what is so far known about the temporal ordering of these. In the present example everything is known; the order is total.

The third part of the time and action map, labelled P identifies the set of actions which have so far been identified. In simple temporal projection problems that set is constant throughout the process; in problems with causation of actions as well as for planning problems the set grows during the process. In the present example the set consists of two actions, as illustrated in the figure.

The fourth part, labelled R, X, Y, Z identifies what is known about the value of each property at each time-point; the value may be T, F, or U. Also

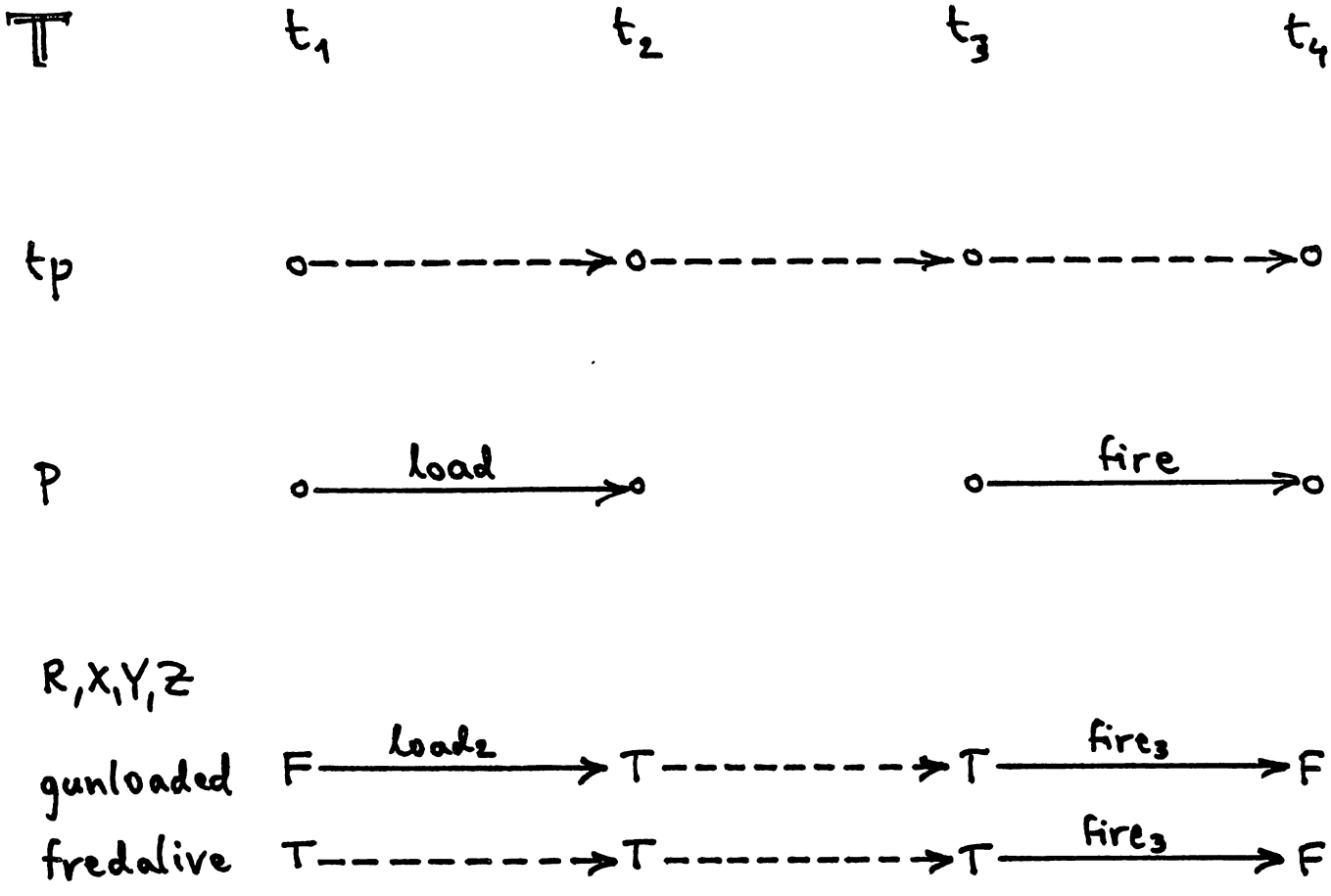
$\mathbb{T}$
t_1
t_2
t_3
t_4
tp
p
load
fire
R,X,Y,Z
gunloaded
fredalive
F
T
$load_2$
$fire_3$

Figure 1.

the fourth part identifies what is known about what properties are controlled by what actions, in what intervals (drawn with a solid arrow), steadied by one or more actions (drawn with a dotted/broken arrow), or is inferred to persist (drawn with a broken arrow). These correspond to the Y, Z, and X components of interpretations, respectively. (If a property is steadied in an interval then it also persists there, but in such cases it would be natural to only draw the arrow representing that it is steadied. Apart from that there can be no possibility of several kinds of arrows in the same interval for a property).

The verification process goes as follows. We initialize the t-a-map as a structure which satisfies all the scene axioms in Γ_s, as illustrated at the top of figure 2. In this example it is only the fourth component which changes, so figure 2 contains a succession of variants labelled a,b,...for the fourth component. Thus e.g. "figure 2c" will refer to the top three components in figure 2, plus the "c" variant of the fourth component.

The t-a-map is initialized to be as in figure 2a. It is easily seen that already with that information, all axioms in Γ_s (the scene axioms) are satisfied, and so are $\Gamma_{t,1}$ which say that the ending-time of an action succeeds the starting-time. We then add to the structure additional information which is necessary there in order that other axioms shall also be satisfied. At the end of the process we shall have one or more t-a-maps which satisfy all the axioms in Γ_a, $\Gamma_{t,2}$ and $\Gamma_{t,3}$ [1] as well.

Given the t-a-map in figure 2a, the obvious choice is to first make sure that Γ_{load} is satisfied for the only instance of its variables which satisfy the antecedent. In the t-a-map in 2a the axiom evaluates to $T \rightarrow F \vee U$ which is U, so more information has to be added. Clearly the first consequence disjunct can not be satisfied, unless some information is removed from the t-a-map, so the only option is to add information so as to satisfy the second disjunct. The result is as in figure 2b, i.e. we recognize that *gunloaded* is T at the end of the *load* action, and is controlled by that action. (The arrow labelled $load_2$ in the figure designates the control, and the index $_2$ is a reminder that the second consequence disjunct was chosen in Γ_{load}).

If we make the optimistic assumption that we will be able to find a maximally <-preferred model, where the abnormal change set $abch(J)$ is empty, and the set P of actions does not grow beyond the present, then we can add some more conclusions: there is no possibility to derive a conclusion of the form

$$Notpersists(t_1, fredalive, t_2)$$

so we can tentatively add to the t-a-map that the property $fredalive$ persists from t_1 to t_2. Also, based on the assumption of an empty $abch$-set, we can conclude that $fredalive$ is T in t_2 as well as in t_1. However, both of these

[1] In this particular example the axioms $\Gamma_{t,3}$ are actually also satisfied right from the start, but that does not hold in general.

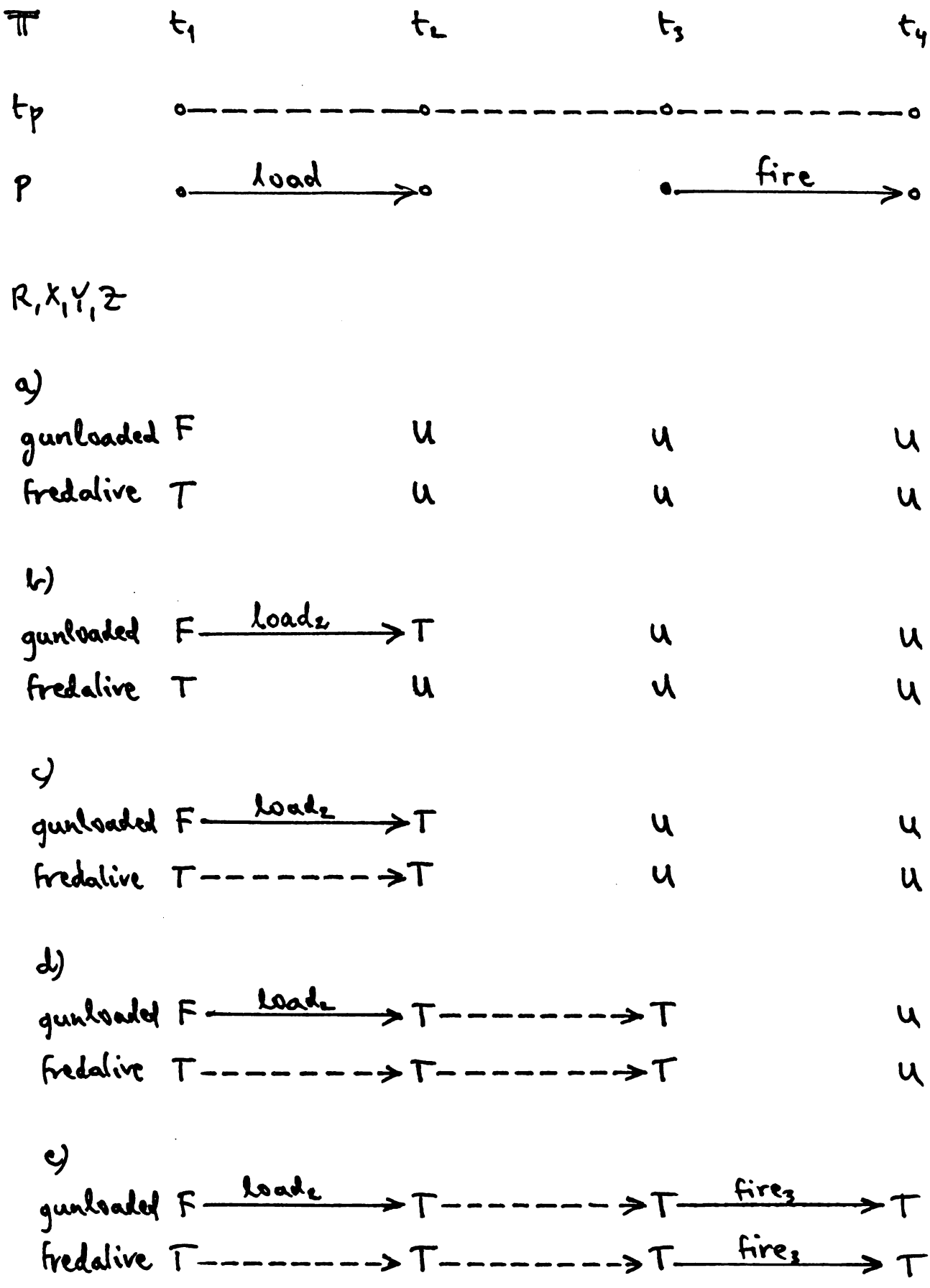
π
t1
t2
t3
t4
tp
p
load
fire
R, X, Y, Z
a)
gunloaded F u u u
fredalive T u u u
b)
gunloaded F load2 T u u
fredalive T u u u
c)
gunloaded F load2 T u u
fredalive T T u u
d)
gunloaded F load2 T T u
fredalive T T T u
e)
gunloaded F load2 T T fire3 T
fredalive T T T fire3 T

Figure 2.

assumptions may later have to be withdrawn, if we have to go to less optimal models. Anyway the working hypothesis at this point is as in figure 2c.

Next, addressing the time interval from t_2 to t_3, we make a similar tentative inference that both *gunloaded* and $fredalive$ persist during that interval, in the absence of any action that would control them and in the absence of any abnormal property change. The result is as in figure 2d.

Finally, in order to satisfy the only relevant instance of axiom Γ_{fire}, we can only proceed to the time-map in figure 2e which of course is the same as we already saw in figure 1. We now have a time-map which can easily be verified to satisfy all the axioms; it is maximally preferred according to $<$ and $\ll$, and it is intuitively fairly clear that all other models are less preferred. (To be precise, of course, the t-a-map is not a single model, but a model schema which can be instantiated into a set of models, which happens to equal the set of maximally preferred models). The conclusion that we are eagerly waiting to arrive at, $\neg Holds(t_4, fredalive)$, is T in the final t-a-map, so it is preferentially entailed by the given axioms.

This example is of course embarrassingly simple, but there is every reason to believe that the method we have seen at work here (and which the reader may recognize as "essentially constraint propagation" or "essentially semantic tableaux") can be applied to significantly larger exercises as well. For more complex examples the procedure must however deal with sets of t-a-maps rather than a single one. First, if the given scene axioms contain disjunctions (such as "either Fred or the turkey chicken Phil is alive") then they already generate a set of alternative t-a-maps which must be processed in parallel. Also, if there is no axiom that can be directly applied in a t-a-map, one has to replace it with several more specific ones, and apply the axiom in each. For example if it is not known whether or not the gun is loaded at time t_1, one has to consider both alternatives in order to be able to apply Γ_{load}.

Similarly for concurrency control: if the axioms state that there are two actions, and it can be deduced that they can not occur concurrently, and the scene axioms does not specify their order, one must split into one case where one action occurs first, and another case where the other occurs first, in order to satisfy $\Gamma_{t,3}$.

The procedure must therefore maintain a *working set* of t-a-maps, and add information to all of them until they all satisfy the axioms. While doing so, it may happen that some of the t-a-maps is not able to satisfy an axiom no matter what information is added to the map. In such a case that alternative is discarded from the working set.

In addition there are the considerations of the preference order $<$, which did not come into play in the present example. Essentially what we have done here is to assume minimality with respect to $<$ and work the above

algorithm.[2] However if the algorithm should get to the point where *no* member remains in the working set, it has to compromise its assumption with respect to $<$, and admit a larger set P of actions, or admit abnormal changes. There again there will be branching, since in principle one can grow P in many ways; it is only the root of that preference structure that is a single node. The procedure must in principle search the tree defined by $<$ from the initial node, and in each direction proceed until it finds a node (hypothesis about P and $abch(J)$) for which the "lower" level of the algorithm concludes with a non-empty working set.

This characterization of the general procedure is easy enough to understand informally, and would presumably be clear enough to serve as a first scetch of an implementable system. What however is not immediately clear is whether such an algorithm would in fact result in a set of time-maps (partial models) with the very desirable property that a proposed conclusion α is T in all those partial models, if and only if it is preferentially entailed by the given axioms according to the definitions given in part I and II of this paper. We will show that that is in fact the case.

3. Semantics of partial interpretations.

We have previously defined the semantics for our temporal logic with total interpretations where logical formulas have the truth-value T or F. In this section we generalize to partial interpretations and a three-valued semantics.

PARTIAL INTERPRETATIONS.

A *partial interpretation* is a tuple $\langle \mathbf{T}^+, tp, R, P, M_T, M_A, X, Y, Z\rangle$, which in general resembles ordinary (total) interpretations, but with the following changes. The old set $\mathbf{T}$ which was totally ordered by tp is now extended into a larger set $\mathbf{T}^+ \supseteq \mathbf{T}$, and tp is redefined as a mapping from pairs of times, to relation symbols:

$$\mathbf{T}^+ \times \mathbf{T}^+ \mapsto \{[\prec], [\preceq], [\succ], [\succeq], [\sim], [\simeq], [\not\simeq]\}$$

The idea is that the members of $\mathbf{T}^+ - \mathbf{T}$ are *time tokens* rather than specific time-points. For example, t_1 through t_4 that were used in the t-a-map in the previous section were such time tokens. (In that particular case we used temporal constant symbols as time tokens, but that is not a necessary choice). The ordering tp expresses the relationships between members of $\mathbf{T}^+$, be they time-tokens or time-points, so that $tp(t, t') = [\preceq]$ means that the current partial interpretation "knows" that $t \preceq t'$, but does not know whether $t \prec t'$ or $t \simeq t'$. Here $t \simeq t'$ represents that they occur at the same actual time-point. The value $tp(t, t') = [\sim]$ represents lack of any knowledge about the order of t and t'.

The ordering tp, now represented in this somewhat nonstandard way, is

[2] We use the term algorithm as a synonym of procedure

still assumed to have the obvious properties of transitivity etc. It partitions $\mathbf{T}^+$ into a set of equivalence sets of times, with $\simeq$ as the equivalence relation. Two different members of $\mathbf{T}$ (the set of time-points) can not be in the same equivalence set, and $\mathbf{T}$ is totally ordered like before.

A strength order $\sqsubseteq$ is defined on tp functions in the obvious way. We define $\sqsubseteq$ first for the temporal relation symbols, as illustrated by the Hasse diagram in figure 3. (An extra top element $\wr$ has been added there to complete the lattice, but may not be used in interpretations). Then $tp \sqsubseteq tp'$ is defined as

$$\forall t \forall u [tp(t,u) \sqsubseteq tp'(t,u)]$$

We define $eqv(t)$ as the equivalence set that t is in, i.e.

$$\{t' \mid tp(t,t') = [\simeq]\}$$

Also we write $ivls([\preceq], tp)$ for the relation on pairs of equivalence sets whose members are pairwise $[\preceq]$, i.e.

$$\{\langle eqv(t), eqv(t')\rangle \mid tp(t,t') \sqsupseteq [\preceq]\}$$

The maximally $\sqsubseteq$-strong tp mappings are those where there is one equivalence set for each member of $\mathbf{T}$, and no other one. In other words, each time-token has then been equivalenced with some time-point.

In practical use of the algorithm, one will presumably only use members of $\mathbf{T}^+ - \mathbf{T}$, since the exact choice of time-point (in $\mathbf{T}$) is irrelevant for normal deduction examples. However we choose to keep time-points and time-tokens in the same set $\mathbf{T}^+$ so that total interpretations can be seen as a special case of the partial ones.

With these conventions for times, most of the other elements of the partial interpretation are easily taken care of:

R is a mapping $ivls([\preceq], tp) \times C \mapsto \{T, F, U\}$

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

MT is a mapping $TC \mapsto \mathbf{T}^+$

MA is a mapping $AC \mapsto P$

X is a mapping $ivls([\preceq], tp) \times C \mapsto \{T, F, U\}$

Y, Z are mappings $P \times C \mapsto \{T, F, U\}$

In the cases of R and X, we have to change the definition that was used for total interpretations (subset of $\mathbf{T} \times C$) since we now have a weaker structure on the set of times, but still want to accumulate the information from axioms (for example ground statements of the form $Holds(t, c, u)$) into the partial interpretation. We request the mapping R to satisfy the following two conditions, and similarly for X:

1) if $R(\langle t, u\rangle, c) = T$ and $R(\langle t', u'\rangle, c) = F$, then tp must be such that either $u \preceq t'$ or $u' \preceq t$;

2) if b is either T or F (but not U), then

$$R(\langle t, t'\rangle, c) = b \wedge R(\langle t', u\rangle, c) = b \leftrightarrow R(\langle t, u\rangle, c) = b$$

In this way R and X can represent current hypotheses about intervals where *Holds* and *Prevails* hold or do not hold.

STRENGTH ORDERING $\sqsubseteq$ ON PARTIAL INTERPRETATIONS.

The strength order $\sqsubseteq$ is defined on partial interpretations and their components as follows. Assume

$$J_1 = \langle \mathbf{T}^+, tp_1, R_1, P, M_{T,1}, M_{A,1}, X_1, Y_1, Z_1\rangle$$

$$J_2 = \langle \mathbf{T}^+, tp_2, R_2, P, M_{T,2}, M_{A,2}, X_2, Y_2, Z_2\rangle$$

If the $\mathbf{T}^+$ components are different then one must permute them in order to compare the two interpretations, and if the P components are different then the two interpretations are unrelated with respect to $\sqsubseteq$. With these components equal we define $J_1 \sqsubseteq J_2$ iff all of the following hold:

$$tp_1 \sqsubseteq tp_2$$

$$R_1 \sqsubseteq R_2$$

$$X_1 \sqsubseteq X_2$$

$$Y_1 \sqsubseteq Y_2$$

$$Z_1 \sqsubseteq Z_2$$

$$M_{T,1} = M_{T,2}$$

$$M_{A,1} = M_{A,2}$$

With the exception of the M components, all the elements which are being compared are mappings. In general we say that $F_1 \sqsubseteq F_2$ for mappings iff

$$F_1(x) \sqsubseteq F_2(x)$$

for all arguments to the mapping. We therefore only need to define $\sqsubseteq$ for the ranges of the mappings being used. We just defined it for the range of tp (figure 3), and already in section 1 we defined it for $\{T, F, U\}$ so that $U \sqsubset T$ and $U \sqsubset F$.

In parts I and II of this paper the M component was defined to be irrelevant for the preference ordering. The reason why we strengthen the requirement here is so that evaluation of formulas will be montone with respect to $\sqsubset$.

For the R and X components there is the additional technical detail that R_1 and R_2 may have different domains, since their domain depends on tp. However it is easily seen that if $tp_1 \sqsubset tp_2$, then $ivls([\preceq], tp_2)$ is obtained from $ivls([\preceq], tp_1)$ by adding more relationships between existing equivalence sets, and/or by merging such equivalence sets, and taking the unions of the relationships going to or from them. (If $tp_1 \not\sqsubseteq tp_2$ then $J_1 \not\sqsubseteq J_2$ anyway). The

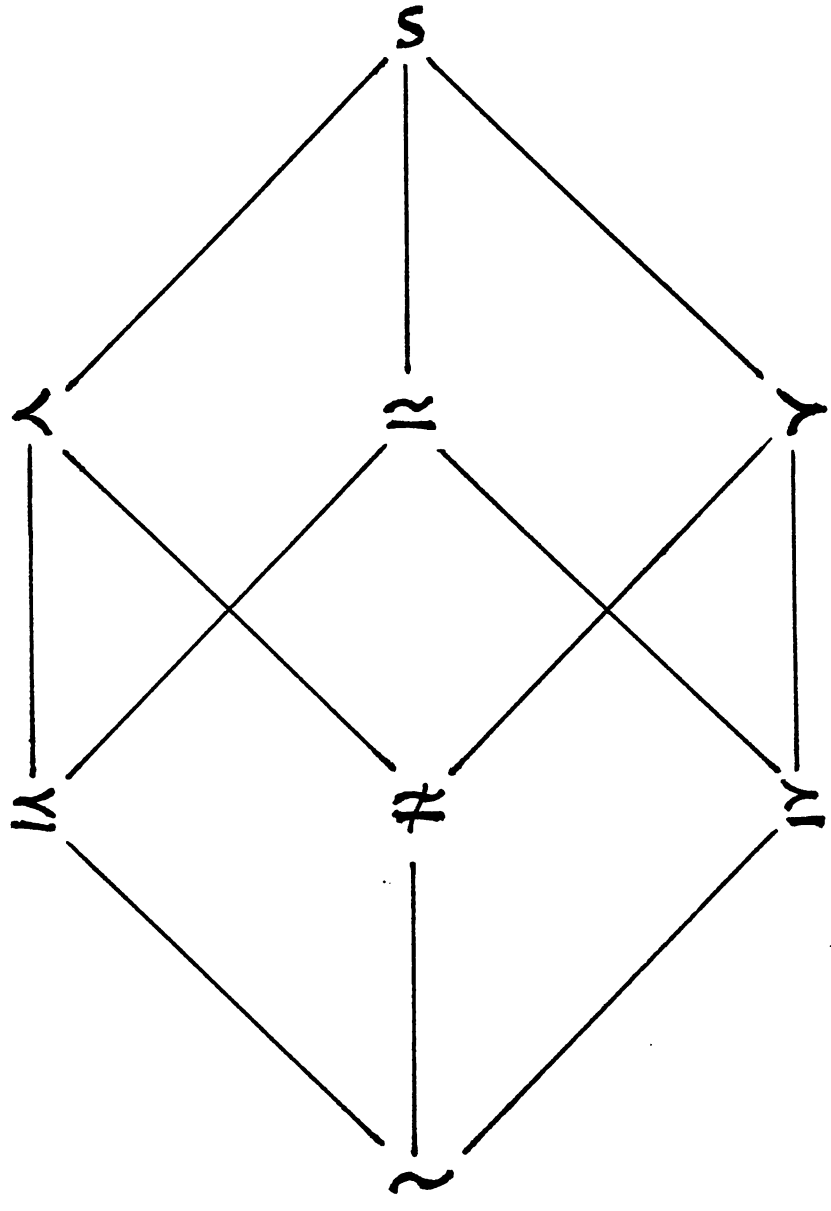

Figure 3.

way to compare R_1 and R_2 for the purpose of determining whether $J_1 \sqsubseteq J_2$ is therefore to apply R_1 to the domain of R_2 in the obvious way, if necessary strengthening it so that the requirements on the structure of R are satisfied, and then determine whether the result is $\sqsubseteq R_2$ on an argument by argument basis.

The elements of the interpretations can be strengthened independently of each other, except that if tp is strengthened then R and X may have to be strengthened too, because of the constraints on them. Also if one element of tp is strengthened, other elements of tp may have to be strengthened by consequence.

When we defined $\sqsubset$ for the relation symbols, we added an extra element $\wr$ in order to complete the lattice at the top, although not allowing it to appear in interpretations. It would in fact be natural to do the same thing for the truth-values, and add a fourth truth-value K such that $F \sqsubset K$, $T \sqsubset K$. With these extensions, and if we take also interpretations containing $\wr$ and K into account, we could expect $\sqsubset$ to define lattices in all domains where we have now defined it. These considerations support using the $\sqsubset$ symbol for the partial ordering on interpretations, but are not directly useful for obtaining the results in the present paper.

EVALUATION OF LOGICAL FORMULAS

Auxiliary definitions: let r and r' be members of the set of temporal relation symbols $\{[\simeq], [\preceq], ...\}$. We define $timplies(r, r')$ as T, F, or U, informally to mean "is it the case that r' if we know r, certainly, certainly not, or maybe?" or formally as

if $r \sqsupseteq r'$ then T

if $r \sqcup r' = \wr$, i.e. the top of the lattice, then F

otherwise U.

For example,

$$timplies([\simeq], [\prec]) = F$$

$$timplies([\preceq], [\prec]) = U$$

Also we define $and(b, b')$ on three-valued truth-values according to Kleene's strong definition [Kle52], i.e. as $min(b, b')$ in an ordering where F is less than U which is less than T. Similarly $not(b)$ is defined by $not(T) = F$, $not(F) = T$, $not(U) = U$.

Now let J be a partial interpretation and α be a logical formula with the syntax defined before. The value of an α in J under the variable assignment VA is written

$$J \models \alpha_{[VA]}$$

and is T, F, or U according to the following definitions.

If α has the form $t = u$: $timplies(tp(val(t), val(u)), [\simeq])$

If α has the form $t \prec u$: $timplies(tp(val(t), val(u)), [\prec])$

If α has the form $Holds(a, t, h, u)$: Let $val(a) = \langle t', h', u' \rangle$. If $h \neq h'$

then the value is F, otherwise

$and(timplies(tp(val(t), t'), [\simeq]), timplies(tp(val(u), u'), [\simeq]))$

If α has the form $Holds(t, c, u)$: $R(\langle eqv(val(t)), eqv(val(u)) \rangle, c)$

If α has the form $Persists(t, c, u)$: $X(\langle eqv(val(t)), eqv(val(u)) \rangle, c)$

If α has the form $Notpersists(t, c, u)$: $not(X(\langle eqv(val(t)), eqv(val(u)) \rangle, c))$

If α has the form $Controls(a, c)$: $Y(val(a), c)$

If α has the form $Steadies(a, c)$: $Z(val(a), c)$

If α has the form $\beta \wedge \gamma$: $and(val(\beta), val(\gamma))$

If α has the form $\neg\beta$: $not(val(\beta))$

If α has the form $\forall v \beta$: using the definition for $\wedge$; universal quantification is seen as the conjunction of the instances for all appropriate choices of value for v.

These definitions immediately give us

COROLLARY.

$$J \sqsubseteq J' \rightarrow (J \models \alpha_{[VA]}) \sqsubseteq (J' \models \alpha_{[VA]})$$

(Remember that $J \models \alpha_{[VA]}$ stands for the value of α in J under VA). This property is the key to the computational method that is to follow below.

Proof. We observe that *not* and *and* are monotone in their arguments, and that

$$b \sqsubseteq b' \rightarrow timplies(b, r) \sqsubseteq timplies(b', r)$$

Since all parts of the definition of formula evaluation have been formed from the interpretation components themselves (including *val* which is directly formed from them), and composed using *not*, *and*, *eqv*,[3] and *timplies*, the result follows by induction.

PREFERENCE ORDER

In our undertaking to generalize the ordinary semantics to work for partial interpretations, it now only remains to generalize the preference orderings. Clearly it would be desirable to say that $J_1 \ll J_2$ for partial interpretations iff every completion of J_1 is $\ll$ every completion of J_2. In fact it is convenient to define $\ll$ so that it is also three-valued, with the value T in the case where *every* completion of J_1 is $\ll$ *every* completion of J_2, and F in the case every J_1 completion is $\not\ll$ every J_2 completion. The same applies for $<$.

Looking back to the definitions of these orderings for total interpretations in part II, we notice first of all that $\ll$ is defined in terms of subset relation-

[3]The verification of monotonicity for *eqv* is omitted

ships on the interpretation components X, Y, and Z. All of them were relations there, and have now been generalized to mappings into $\{T, F, U\}$, i.e. to three-valued relations. Since $\subset$ between relations is essentially an implication, we generalize into a relation $\breve{\subset}$ between $\{T, F, U\}$-valued relations, which is a corresponding implication in the three-valued logic.

The definitions are simple:

If b and b' are in $\{T, F, U\}$, $b \breve{\subseteq} b'$ is defined as $or(not(b), b')$, where *or* is defined as a *max* function on the ordering where *and* was previously defined to minimize. The truth-table for this operation is seen in figure 4. For definite (non-U) truth-values, this table is like ordinary material implication, and for U arguments it has the strongest $\sqsubset$-monotone definition.

If F and F' are three-valued functions over the same domain, $F \breve{\subseteq} F'$ is defined as the minimum (in the same sense as for *and*) of $F(x) \breve{\subseteq} F'(x)$ for all x. Also $F \breve{\subset} F'$ is defined as $and(F \breve{\subseteq} F', not(F' \breve{\subseteq} F))$, and $F \breve{=} F'$ as $and(F \breve{\subseteq} F', F' \breve{\subseteq} F)$.

With these definitions we can transfer the definition of $\ll$ essentially as it is, just cupping all the relations that compare three-valued relations, and adding the requirement that *tp* is equal (which was automatic before):

$$
\begin{aligned}
J_1 \ll J_2 == {} & tp_1 \breve{=} tp_2 \wedge R_1 \breve{=} R_2 \wedge P_1 = P_2 \wedge \\
& ((Y_1 \breve{\subset} Y_2 \wedge Z_1 \breve{\subseteq} Z_2) \vee \\
& (Y_1 \breve{\subseteq} Y_2 \wedge Z_1 \breve{\subset} Z_2) \vee \\
& (Y_1 \breve{=} Y_2 \wedge Z_1 \breve{=} Z_2 \wedge X_1 \breve{\supset} X_2))
\end{aligned}
$$

We assume that $\wedge$ and $\vee$ in this formula are evaluted in the three-valued way.

COROLLARY. The three-valued relation $\ll$ is $\sqsubset$-monotone in both arguments.

This follows since the definition of $\ll$ is entirely composed of $\sqsubset$-monotone functions. (The ordinary equality that is used for comparing P in the definition does not matter, since interpretations that compare with $\sqsubset$ must have equal P components anyway).

COROLLARY. If $I \ll I'$ is T for two partial interpretations I, I', and $I \sqsubseteq J$, $I' \sqsubseteq J'$, then $J \ll J'$ is T.

Next we proceed to extending the definition of the other ordering, $<$, which minimizes the set of abnormal changes and the set P of actions. However, unlike for $\ll$, the ordering $<$ for non-total interpretations is not used in any essential way in the description and analysis of the decision procedure in the next section. The definitions that follow are therefore mostly for symmetry and completeness.

The criterium on P need not change at all, compared to the definition for total interpretations, since our definition of partial interpretations does not

Truth table for $b \mathrel{\breve{\subseteq}} b'$

b \ b'	T	U	F
T	T	U	F
U	T	U	U
F	T	T	T

Figure 4.

have any partiality with respect to whether actions occur or do not occur. However we must define what is the set $abch(J)$ of abnormal changes in a partial interpretation.

The appropriate definition is for $abch(J)$ to be a mapping

$$ivls([\preceq], tp) \times C \mapsto \{T, F, U\}$$

which will assign to every interval (between equivalence classes of times) the information that the property c has at least one abnormal change there (T), does not have any abnormal change (F), or may have an abnormal change depending e.g. on how tp is strengthened (U). With this definition it is straightforward to compare abnormal change sets using $\breve{\subseteq}$, and the definition for $<$ can be transferred from before as follows:

$$J_1 < J_2 == [abch(J_1) \breve{\subset} abch(J_2)] \vee [abch(J_1) \breve{=} abch(J_2) \wedge P_1 \subset P_2]$$

The definition of $abch(J)$ then is the three-valued function $A(w, c)$ (where w is a pair of equivalence sets of times) which is T iff $X(w, c) = T$ (meaning we expect persistence during the interval w) and there are two subintervals w' and w'' necessarily overlapping with w according to tp, such that $R(w', c) = T$ and $R(w'', c) = F$. The definition for when the function is U or F will not offer any difficulty.

It seems likely that $<$ is also monotone with respect to $\sqsubseteq$ but we do not have any result on that point. The correctness proof for the algorithm in the next section needs the monotonicity of $\ll$ but not of $<$.

With this we conclude the generalization from ordinary to partial interpretations. Clearly the generalization is quite straight-forward, and when one has become used to the three-valued calculus it is actually no more complex than the original definitions.

4. Decision procedure.

In section 2 we described a plausible verification procedure through a simple example. Let us now go back and analyze that example with the conceptual tools that have been introduced in the previous section, and meanwhile generalize into a verified semantic decision procedure.

The specification for the procedure is to determine whether a certain Γ entails a certain α, i.e. whether α has the value T in all members of

$$Min_{<}(Min_{\ll}(Mod(\Gamma)))$$

Since in this section we will work with such nestings of monary functions, let us decide at once to omit parentheses and write the specification as

$$Min_{<}\ Min_{\ll}\ Mod\ \Gamma$$

Here Mod is the function which takes the set of all total models (no partial models). We shall call this the *specification formula,* and its value the *speci-*

fication set. Γ is assumed to consist of the three types of axioms defined at the end of part II of the paper sequence.

The computational problem of course is that this set is too large, and that in many cases the differences between its members are irrelevant for the value of α. The whole idea with introducing partial interpretations was that we should be able to replace the specified set with a smaller set of partial interpretations, which still give the same result when α is evaluated over them.

We shall use the term *test set* for the computed set of interpretations where α is to be evaluated. The specification of correctness for the algorithm can then equivalently be phrased as whether α is T in all members of the test set, iff it is true in all members of the specification set. In this section we need to discuss (1) what is a correct choice for the test set, and (2) what is an algorithm for computing the test set. Let us start with the first question, although in the continued discussion we will cover both questions in parallel.

The use of partial interpretations immediately suggests that the test set should be something like

$$F^* \; Min_{<} \; Min_{\ll} \; Mod \; \Gamma$$

where F is a function which takes a set $\Delta \cup \Delta'$ of interpretations and replaces it by $\Delta \cup \{J'\}$, where J' is a partial interpretation which is $\sqsubseteq$ every member of Δ', and F^* performs the F operation repeatedly. Every definite value (T or F) obtained with the new test set is clearly also a value for the original test set. Such a test set is therefore sound, although not complete since there is no lower bound to the choice of J'.

One question which can come up for this or any other choice of test set is what happens if α has the value U in some member of the test set. In such a case one must replace that element in the test set by several $\sqsubset$-larger ones, and evaluate α in them instead. (Since α has to be T in all of them in order to be entailed, it is sufficient to compute until we have found one where α is F). Rather than having to assume that the choice of test set shall depend on α, we make the convention that to "evaluate α in the members of the test set" shall be considered to also include splitting test set members into several $\sqsubset$-larger ones whenever made necessary by α.

A stronger choice of F than the one just mentioned, is to take the set of all partial interpretations J which are such that the set of completions of each J is a subset of the specification set, and which also are $\sqsubseteq$-minimal with that property. This choice will clearly always give the right result, if combined with the described policy when α evaluates to U in a member of the test set. It is something of this kind that we would like as a test set, but of course we have to compute it in a different way. Let us discuss the computational process some before defining the test set precisely.

As we saw in section 2, the strategy for the procedure is to start with a partial interpretation which is a model for *some* of the axioms, and to

gradually strengthen it in the sense of $\sqsubset$, and sometimes also in the sense of $<$, namely when additional members of P or some members of the abnormal change set have to be introduced. Notice however that the two orderings $\ll$ and $<$ relate differently to $\sqsubset$. The $\ll$-preferences are realized while going to $\sqsubset$-larger partial interpretations, since we start with a p.i. where X, Y, and Z map all argument combinations to U. In the course of deduction we will strengthen some components of Y and Z to T, and we can minimize by strengthening the remaining components from U to F. Similarly the components of X are default-strengthened from U to T since we maximize the relation *Persists*.

For the $<$ relation, on the other hand, decisions to compromise on $<$ (going to a $<$-larger p.i.) by introducing additional actions takes us to partial interpretations which are *not* $\sqsubset$-stronger. Allowing for abnormal changes again can be done within $\sqsubset$, in the same manner as for $\ll$ strengthenings, but they are of course the last resort.

As we saw in the example, the difference between the two orderings is also reflected in the procedure. In the example we made the assumption that a minimum with respect to $<$ could be achieved, and the computations were then done under that assumption. For more difficult problems one needs an algorithm in two layers, as was outlined towards the end of section 2. Let us now translate that algorithm into more formal terms which are needed in order to pursue the analysis. We define the algorithm in two ways, one simpler definition which essentially defines away the outer layer and only specifies the inner layer (which is what we saw at work in the concrete example in section 2), and a more precise, two-layer definition which accounts for both layers.

THE ONE-LAYER DECISION PROCEDURE.

The procedure operates on a set of *hypotheses,* each of which is a pair $\langle J, \xi \rangle$, where J is a partial interpretation and ξ is the kind of mapping which is used as the value of $abch(J)$, but mapping only to definite truth-values, i.e. a mapping

$$ivls([\preceq], tp) \times C \mapsto \{T, F\}$$

where tp is taken from J. The algorithm first determines a set Δ of partial models for the axioms in $\Gamma_s \cup \Gamma_{t,1}$ which is complete for those axioms in the sense that every total model of the axioms is $\sqsupseteq$ some member of Δ. The members of Δ do not have to be $\sqsubset$-minimal, but of course we expect them to be at least close to minimal. Technically we write this requirement as

$$Mod\ (\Gamma_s \cup \Gamma_{t,1}) \ = \ Compl\ \Delta$$

where *Compl* is a function which takes a set of partial interpretations and replaces each of them by all total interpretations $\sqsupset$ to it. Since $\Gamma_s \cup \Gamma_{t,1} \subset \Gamma$, we then have

$$Mod\ \Gamma \ \subseteq \ Compl\ \Delta$$

The real, two-layer procedure which we will introduce below will pick out a few, preferred members of Δ, but the present more abstract algorithm considers them all. (They are of course infinitely many). Consider now the set of hypotheses consisting of all pairs

$$\langle J, \xi \rangle$$

where $J \in \Delta$, and ξ is any mapping as defined above which fits the *tp* component of J. We write the pair formation operation as $Pairxi$, taking a set of p.i:s to a set of pairs, so the set of initial hypotheses can then be written as

$$Pairxi\ \Delta$$

The algorithm now operates by successively removing members of the working set, modifying them, and replacing no, one, or more new members to the working set. Each time it is only the J component that changes; the ξ component is unchanged. Also the change in J always consists of replacing it by some J' such that $J \sqsubseteq J'$.

The actual changes which are done to J are those which (1) change it so as to satisfy the other axioms, (2) minimize with respect to $\ll$, and (3) draw the conclusions from the ξ component specifying absence of abnormal change. Thus if the current J specifies that a property c persists over a time interval w, and $\xi(w, c) = F$ meaning no abnormal change there, then the process may propagate values across the interval. If the value of c at one end of w is T and at the other end is U, then one can increase it to T at the other end. If it is T at one end and F at the other end, then the hypothesis is discarded.

The result of the process just described shall be written as

$$Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$$

where $Enforce$ is the operation which does $\sqsubset$ increases until all axioms are satisfied, and $Prop$ propagates property values according to the current ξ.[4] We assume that the operation $Enforce$ has the property (for arbitrary set Δ' of partial interpretations) that

$$Compl\ Enforce\ \Delta' = Enforce\ Compl\ \Delta'$$

The operation $Enforce$ in the case of total interpretations of course only serves as a filter, to remove interpretations which do not satisfy all the axioms, since there is no more information to be added. What this equation says is therefore that the $Enforce$ operation on partial interpretations does not "lose" any significant hypotheses along the way to stronger p.i:s.

Also, since $Enforce$ serves as a filter on total interpretations, checking that they are models for all the axioms, we have

$$Enforce\ Compl\ \Delta' \sqsubseteq Mod\ \Gamma$$

[4]The three operations $Enforce$, $Min_{\ll}$, and $Prop$ are defined for hypotheses as applying to their first element while leaving the second element unchanged

if Δ' is an arbitrary set of partial interpretations.

The abstract algorithm runs the $Enforce$, $Min_{\ll}$, and $Prop$ operations in succession, and then minimizes with respect also to the $<$ ordering. Thus it can be summarily written as the *computation formula*

$$Minx_{<}\ Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$$

where of course $Min_{\ll}$ takes a set of hypotheses and removes all the non-minimal members of the set. (Some fine points of that operation will be discussed in the course of the proof below). We write $Minx_{<}$ rather than $Min_{<}$ to emphasize that $Minx$ refers to the explicit ξ component of the hypothesis as the basis for the preference ordering, whereas $Min_{<}$ uses $abch(J)$ i.e. computes abnormal changes directly from the partial interpretation.

The value of this computation formula is a set of surviving hypotheses, which we have already agreed to call the *test set.* This completes the description of the simple algorithm. END OF ALGORITHM.

Obviously what the next version of the procedure will do is to make sure that we do not operate on all the infinitely many hypotheses at once, and instead to initially only consider the $<$-preferred ones. Also later on we shall accomodate the fact that the $Enforce$, $Min_{\ll}$, and $Prop$ operations should be interleaved rather than be done in succession. However let us first prove the correctness of this abstract, one-layer procedure.

The correctness criterium for the algorithm is whether α evaluates to T in all members of the test set iff it evaluates to T in all members of the specification set However this criterium can immediately be replaced by modifying the computation formula so as to also take the completion of its result, and removing the ξ component, obtaining the *completed computation formula*

$$Unpair\ Compl\ Minx_{<}\ Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$$

whose value is the *completed test set.* The operation $Unpair$ extracts the first element of each hypothesis, and discards the ξ component introduced by $Pairxi$. What we then have to determine is whether the completed test set equals the specification set, according to the specification formula

$$Min_{<}\ Min_{\ll}\ Mod\ \Gamma$$

The strategy for proving that equality is to show that each of the following transformations preserves the value of the expression:

0. $Unpair\ Compl\ Minx_{<}\ Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$
1. $Unpair\ Minx_{<}\ Compl\ Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$
2. $Unpair\ Minx_{<}\ Prop\ Compl\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$
3. $Unpair\ Minx_{<}\ Prop\ Min_{\ll}\ Compl\ Enforce\ Pairxi\ \Delta$
4. $Unpair\ Minx_{<}\ Prop\ Min_{\ll}\ Compl\ Pairxi\ Enforce\ \Delta$

5. $Unpair\ Minx_{<}\ Prop\ Min_{\ll}\ Pairxi\ Compl\ Enforce\ \Delta$

6. $Unpair\ Minx_{<}\ Prop\ Min_{\ll}\ Pairxi\ Mod\ \Gamma$

7. $Unpair\ Minx_{<}\ Prop\ Pairxi\ Min_{\ll}\ Mod\ \Gamma$

8. $Min_{<}\ Min_{\ll}\ Mod\ \Gamma$

We now verify each of these steps in succession, by proving the lemma that is instrumental for performing the step. Meanwhile we also make more precise the definitions of some of the operations. Throughout Λ stands for a set of hypotheses.

STEP 1. $Compl\ Minx_{<}\ \Lambda\ =\ Minx_{<}\ Compl\ \Lambda$

The operation $Minx_{<}$ is only affected by the P component of the interpretation, and the ξ part of the hypothesis. The operation $Compl$ does not change these components.[5]

STEP 2. $Compl\ Prop\ \Lambda\ =\ Prop\ Compl\ \Lambda$

This holds under the assumption that all propagations can be realized in the partial interpretations, i.e. in the case of $Compl\ Prop\ \Lambda$. More specifically, if the current ξ specifies that a property c shall be invariant over an interval, and J assigns U to c in several sub-intervals, then propagation must be understood as strengthening into two p.i:s, one where c is T in both, and one where c is F in both. On the other hand if it is U in only one of them, propagation just results in one strengthening, and if it is U in neither (for example, in the non-partial interpretation) it works as a filter. With this assumption about the propagation operator, the two operators $Compl$ and $Prop$ do clearly permute.

STEP 3. $Compl\ Min_{\ll}\ \Lambda\ =\ Min_{\ll}\ Compl\ \Lambda$

This is more complex than for step 1, since the relation $\ll$ is not independent of $\sqsubset$. Let us consider what $Min_{\ll}$ really does. It must be understood to perform the following two phases:

1) Minimize with respect to $\ll$ within each partial interpretation. This is done by modifying its X, Y, and Z components so that all arguments which were mapped to U, are instead mapped to T in the case of X, and to F in the case of Y and Z. (It is easily seen that with our choice of axioms in Γ_t, there is no need to do any additional inference between minimization of Y and Z, and maximization of X).

2) In the new working set of p.i:s, select those which are $\ll$-minimal and discard the rest, i.e. if $J \ll J'$ has the value T then J' is discarded. If neither $J \ll J'$ nor $J' \ll J$ is T, then neither can be discarded on this basis. Since we have defined the $\ll$ ordering to be three-valued, we must also consider what happens if the value is U. However by inspection of the definitions we

[5] If we later generalize to axioms that imply existence of actions, then we should probably also revise $<$ so that it minimizes the set of unimplied actions.

see that that cannot happen after phase (1) has been performed, because X, Y, and Z then map all argument combinations to definite values. The truth-table in figure 4 has definite values in the four corners. We easily verify that the new definition of $\ll$ does give the value T to the relation exactly when the preference should apply.

With this more precises definition, the lemma is verified as follows. Consider first phase (1), which transforms a p.i. J into another p.i. J' which is stronger in terms of both $\sqsubset$ and $\ll$. It is clear that all members of $Compl(J) - Compl(J')$, i.e. those completions which were implicitly de-selected in phase 1, are $\gg$ any member of $Compl(J')$ so they were correctly de-selected. Also it is clear that all members of $Compl(J')$ are unrelated w.r.t. $\ll$, since their X, Y, and Z components are equal. Therefore $Compl(J')$ does not contain any members which are non-minimal with respect to $\ll$.

Then considering phase (2), we notice that if J and J' are two different members of the working set where phase (1) has been performed, $J \ll J'$ is either T or F. From the monotonicity of $\ll$ it then follows that $Compl$ and $Min_{\ll}$ permute. This concludes the proof of the lemma for step 3.

STEPS 4 AND 5 are trivial (if one deals like before with the technical detail of revising ξ correctly if $Enforce$ modifies tp).

STEP 6. $Mod\ \Gamma\ =\ Compl\ Enforce\ \Delta$

According to the assumption on $Enforce$ we have

$$Compl\ Enforce\ \Delta\ =\ Enforce\ Compl\ \Delta$$

The latter expression is $\supseteq Enforce(Mod(\Gamma))$ which in term equals $Mod(\Gamma)$, since $Compl(\Delta) \supseteq Mod(\Gamma)$ by the requirements on Δ and since $Enforce$ is clearly $\subseteq$-monotone. But the latter expression is also $\subseteq Mod(\Gamma)$ since $Enforce$ filters a set of total interpretations for satisfying all the axioms in Γ.

STEP 7 is again trivial.

STEP 8. If Δ' is a set of total interpretations, then

$$Unpair\ Minx_{<}\ Prop\ Pairxi\ \Delta'\ =\ Min_{<}\ \Delta'$$

This says that $<$-minimizing total interpretations J using $abch(J)$, is equivalent to forming all pairs $\langle J, \xi \rangle$, then retain those pairs where $abch(J) \breve{\subseteq} \xi$ (which is what $Prop$ does when J is a total interpretation), then $<$-minimize using the ξ component, and then throw away the ξ. - This concludes the proof for the correctness of the one-layer decision procedure.

THE TWO-LAYER DECISION PROCEDURE.

The simplified, one-layer procedure whose correctness we have now proven, says nominally that the whole set Δ of initial partial models for $\Gamma_s \cup \Gamma_{t,1}$ are to be considered in parallel. It therefore does not give any explicit cue to the strategy of starting with those members of Δ which are minimal with respect to $<$, and going to $<$-larger hypotheses only as required.

A notationally quite simple modification of the previous formula does however capture that refinement. Recall that the single layer computation formula was

$$Minx_{<}\ Prop\ Min_{\ll}\ Enforce\ Pairxi\ \Delta$$

apart from the initial *Unpair Comp* which were needed for relating to the specification formula. We now introduce an operation $Partn_{<}$ which partitions a set Λ of hypotheses into a set of equivalence sets under $<$, or more correctly, the members of an inner set are those hypotheses which have the same ξ component, and whose interpretations have the same P component. Also we introduce a reverse operation *Flatten* which takes a set of sets, and formes the union of all the inner-level sets. Thus

$$Flatten\ Partn_{<}\ \Lambda\ =\ \Lambda$$

Now we rewrite the computation formula as

$$Flatten\ Minx_{<}\ Prop^{\bullet}\ Min^{\bullet}_{\ll}\ Enforce^{\bullet}\ Partn_{<}\ Pairxi\ \Delta$$

where $F^{\bullet}$ on a set of sets is defined as applying F to each inner set, and forming the set of the results.

This formula now captures concisely the two-level procedure that was described at the end of section 2. The outer layer is seen as the operation $Minx_{<}$ which ranges over the set $Partn_{<}(Pairxi(\Delta))$, in order to identify those of its members which are still non-empty after the operations *Enforce*, $Min_{\ll}$, and *Prop* (i.e. after the inner layer operations), and which are $<$-minimal with that property. It does so by starting with the $<$-minimal members regardless of the property, trying them, and proceeding to $<$-larger sets if required.

On the other hand, this formula does not capture performance related issues such as search strategies, or issues whether deductions which have been made in one run of the inner layer, can be rescued when the outer layer resumes. That must however be a later step in the analysis. The first steps are to verify the correctness of the above computation formula, and to consider the performance related issues for the inner layer i.e. for the simpler formulation of the algorithm.

The correctness proof goes in two simple steps. We observe that by the construction of the new operators,

$$Flatten\ Minx_{<}\ Partn_{<}\ \Lambda\ =\ Minx_{<}\ \Lambda$$

for an arbitrary set of hypotheses Λ. Also we see that if a function F on sets of hypotheses does not influence the criterium used for deciding the ordering $<$ in those hypotheses, then

$$F^{\bullet}\ Partn_{<}\ \Lambda\ =\ Partn_{<}\ F\ \Lambda$$

Since all three of the lower-layer operations *Enforce*, $Min_{\ll}$, and *Prop* have this property, the result follows.

Notice that throughout this proof we rely on the fact that the $<$ relation as used by $Minx_<$ and $Partn_<$ uses the explicit ξ component in the hypotheses, not the computed value $abch(J)$. The latter is not independent of e.g. the $Enforce$ operation.

The results so far in this section can be summarized as a the following theorem

CORRECTNESS OF DECISION PROCEDURE: In order to determine whether

$$\Gamma \models_{\ll,<} \alpha$$

where Γ has the structure defined earlier here, one obtains the correct answer by the following steps:

1) Select Δ as a set of partial interpretations satisfying

$$Mod(\Gamma_s \cup \Gamma_{t,1}) \;=\; Compl\ \Delta$$

Define Λ as the (infinite) set of sets

$$Partn_< \ Pairxi\ \Delta$$

which is ordered by $<$ and has minimal members (no unbounded descending chains). Compute the minimal members explicitly.

2) Search upwards along $<$ from those minimal members, and compute Φ as

$$Minx_< \ Prop^\bullet \ Min^\bullet_\ll \ Enforce^\bullet \ \Lambda$$

3) Determine whether α evaluates to T in all members of $Flatten(\Phi)$.

5. Deductive control of the minimization search.

Let us now consider the problem of interleaving the operations $Enforce$, $Min_\ll$, and $Prop$. The computation formula as it currently stands does not correctly characterize the order in which things ought to be done. The formula says that the three operations shall be taken in global sequence. Yet in the computational example in section 2, we did not first arrange to satisfy all the axioms, and then minimize $Causes$ and maximize $Persists$, and then again propagate unchanged properties. Instead we interleaved these operations: first we applied an axiom (namely for the load action), then we minimized $Causes$ over the interval of that action, then we maximized $Persists$ over the same interval, then we propagated the property of $fredalive$, and only later did we apply the axioms that were relevant for subsequent time intervals.

Thus the ordering suggested by the computation formula might better be applied locally, to one interval of time, rather than to the whole p.i. We shall now analyze whether such interleaving preserves the correctness of the decision procedure.

The first question to be asked then is whether there is any problem at

all: maybe the later steps $Min_{\ll}$ and *Prop* permute with *Enforce*. Unfortunately they do not, which is easy to see by referring to the concrete example.

For example, before we have applied the axiom for *load*, we have not yet derived that the load action controls the property *gunloaded*. In the current p.i., $Y(w, gunloaded)$, where w is the time period for loading the gun, has the value U because nothing has been deduced or assumed yet. If at this point we make a default assumption that it is F, we go to an interpretation which is $\ll$-smaller, so this is a correct step as a part of $Min_{\ll}$ as we defined it above. The problem is however that as we go $\sqsubset$-upwards from the new p.i. we can never satisfy all the axioms, since in order to satisfy the axiom for *load* we must change $Y(w, gunloaded)$ from F to T which is not possible. Therefore because of the default step this member of the working set will be discarded, whereas if we had not made the default step it would not have been discarded.

There are two conclusions. First, whenever (and if) we proceed in the $\ll$ direction without having first satisfied all the axioms (as we did as a matter of common sense in the example), we should see that step as a backtrack point which we may later revert to, if we can not satisfy the axioms. Secondly, although such $\ll$-decreasing steps can reduce the computation considerably if they are chosen well, they may also have the opposite effect namely if they require backtracking later.

Certainly backtracking is no novelty and no surprise in the context of non-monotonic reasoning, but still there is something important missing here. In the example in section 2 we did not foresee any possibility that backtracking would later be needed. We had drawn all potential conclusions for those actions which could influence the time period from t_1 to t_2, and therefore we could quite safely go ahead with the selective application of the preference relation. With our informal understanding of what goes on, there is no need to foresee any backtracking, as long as we stay with the current ξ.

We propose to capture that informal notion in the formal system by extending the set Γ with rules which control the computation process, making it

$$\Gamma_s \cup \Gamma_c \cup \Gamma_a \cup \Gamma_t$$

The new set Γ_c contains logical formulas which have been obtained as *conclusions* from the other ones, i.e. they are not given by the "user" in the specification. The purpose of the members of Γ_c, called *constrainers*, shall be to selectively perform minimization steps when the conditions are ripe.

One example of such a constrainer could be the following formula, which we shall later refer to as κ:

$$Holds(a, t, load, u) \rightarrow \neg Controls(a, fredalive).$$

which says that a *load* action does not control the *fredalive* property. Use

of such a rule effectively shifts a part of $\ll$-minimization from the nominal minimization step, to an earlier use of a rule during the $Enforce$ step. Another constrainer could say that if all actions (meaning: quantifying over the P component in the interpretation) either are strictly before or strictly after the present time interval, or do not control the present property, then the present property persists over the present time interval.

Such constrainers capture the "control knowledge", using a popular term, that we used in the example in section 2. It is easy to see how the algorithm that was described and analyzed above would operate more efficiently if such constrainers were added to the set driving the computation, but without changing the final result.

Of course, adding more rules to a rule-based system (which in a sense is what we have here) raises questions of how to control their application order. However the algorithm described here does not branch because at some point there are several different rules which may be applied; it only branches when a rule requires several different strengthenings to be considered. The constrainer rules that have been mentioned do not cause any such split; they will only add some more information to the current p.i. and then be satisfied, if the preconditions were T. (If the preconditions of the rule are U and the consequence is not T, the rule must be kept on hold, and if they are F the rule is of course satisfied without any changes in the p.i.)

If one uses the quite natural search strategy to give preference to rules that do not split the current p.i. into several, but just adds information to it, then constrainers will naturally get high priority. In the particular case of the example in section 2, the axiom for *load* will also have high priority since the initial p.i. has enough information to select which of the consequence disjuncts to use, but the axiom for $fire$ will have lower priority at the beginning of the computation because it is undetermined which of its disjuncts should be used. If one has to apply the $fire$ axiom early, one therefore obtains a split into several new members of the working set.

If however the constrainers are available, they will take precedence over the axiom for $fire$, and add to the p.i. the values of *gunloaded* at t_3 and $fredalive$ at t_2 and t_3. Thus they change the p.i. so that the $fire$ axiom also obtains first priority.

In summary, with a limited set of constrainers, and a simple and natural search strategy, our algorithm obtains the common-sense behavior for the section 2 example. Remaining question: how do we obtain the constrainers?

That question breaks down into two parts: how do we prove that a proposed constrainer is correct, and also, how do we identify a set of computationally useful constrainers. We only address the first of those issues.

The natural way to obtain a constrainer would be by logical deduction from the general axioms, i.e. Γ_a and Γ_t. We do not want the constrainers to depend on the scene axioms Γ_s. Is there a way that the given axioms *entail*

for example the constrainer κ that was introduced above?

The specification axiom for the action *load*, already called Γ_{load} does not entail κ by ordinary entailment. On the other hand we do have

$$\Gamma_{load} \models_{\ll,<} \kappa.$$

actually quite trivially since preferred models for Γ_{load} have an empty set P. That however is fairly useless exactly since the entailment is non-monotonic. For any set of axioms which entails κ preferentially but not conventionally, we can always find a superset which does not entail it, for example by explicitly stating a counterexample:

$$Holds(a_2, t_1, load, t_2).$$

$$Controls(a_2, fredalive).$$

which overrides the minimization obtained by $\ll$. In each application we will add a new set of scene axioms, so when a non-monotonic logic is used we can never rely on a derived rule.

However, it is not *intended* that the relation *Controls* should be used explicitly in scene axioms. That relation was only intended for use in the "system axioms" Γ_a and Γ_t. This immediately suggests a revised notion of entailment for use here, defined as follows.

Let L_s be a subset of the language of logical formulas, informally understood as the scene language (whence the index s). Also let Γ, α, $<$ etc. be as usual. We shall say that *across* L_s, Γ *preferentially entails* α, also written

$$[L_s] : \Gamma \models_< \alpha$$

iff, for an arbitrary set of logical formulas Υ,

$$\Upsilon \subseteq L_s \Rightarrow \Gamma \cup \Upsilon \models_< \alpha$$

In particular, given that our applications will use a fixed set Γ_{at} of "system" axioms ($\Gamma_a \cup \Gamma_t$), and a variable set of scene axioms (Γ_s) which must be taken from the sub-language L_s, then every formula κ which satisfies

$$[L_s] : \Gamma_{at} \models_{\ll,<} \kappa$$

is known to hold in all models, in all axiom sets which can be considered.

We must take care, of course, since the constrainer κ itself is not a member of the sub-language L_s. However, preferential entailment does satisfy cumulative monotony and transitivity [Gab85,Mak88], i.e.

$$\Gamma \models_< \alpha \Rightarrow [\Gamma \models_< \beta \Leftrightarrow \Gamma \cup \{\alpha\} \models_< \beta]$$

so adding a set Γ_c of constrainers to the rules driving the reasoning algorithm does not influence the model set.

We can summarize the key property of constrainers as follows:

CONSTRAINER INVARIANCE. If

$$[L_s] : \Gamma_{at} \models_{\ll,<} \kappa$$

and $\Gamma_s \subseteq L_s$, then

$$\Gamma_s \cup \Gamma_{at} \models_{\ll,<} \alpha \quad \Leftrightarrow \quad \Gamma_s \cup \{\kappa\} \cup \Gamma_{at} \models_{\ll,<} \alpha$$

This concludes the analysis of the semantic decision procedure in terms of the concepts and results of the earlier sections. The procedure has been proven correct relative to the specification that derived from parts I and II of this paper sequence. Also it has been shown that the common-sense behavior illustrated in section 2, can be obtained by this procedure, using a natural search strategy, and logically well founded constrainer rules. The constrainer rules contain "control knowledge" in the sense that they do not influence the results of the computation, but they can serve to reduce the number of parallel hypotheses which have to be considered and discarded, and thus help the process to go straight to the solution and to minimize the necessary computation.

6. Discussion.

PARTIAL INTERPRETATIONS. Partial interpretations were introduced here as a way of representing the current state of a semantic tableau, or as an abstraction of the current state of the process in the inner layer of the verification procedure. There is however also a potential other reason for using partial interpretations, namely for an approach to the *qualification problem.*

The qualification problem is well known as the problem of representing the many various exceptional conditions which may prevent an action from happening at all, or from having its usual effects. It is one of the standing unsolved problems of temporal reasoning in A.I. It can be assumed however that the qualification problem requires the use of *defaults,* i.e. rules about what "usually" is the case, and which can be assumed in the absence of information to the contrary.

In the context of our two preference relations $\ll$ and $<$, we want qualification defaults to be made within the $\ll$ ordering or between the two orderings, but not in the $<$ ordering. This is because as is often observed, we expect a planner to preferentially produce a simpler plan (with fewer actions) which will work under the assumption that the qualification defaults hold, and to give less preference to a more complicated plan which accounts for all the defaults. This suggests looking for ways of extending the ordering $\ll$ so as to also account for defaults and qualification.

Reasoning with defaults has an immediate affinity with temporal reasoning in the sense that both types of reasoning is non-monotonic, and can be adequately thought about in terms of preference relations on interpretations. There is however also a significant difference: in the case of temporal reasoning the preference order can be given once and for all (at least according to current thinking), whereas for defaults there is a wide range of special rules

each of which specifies some aspect of the preference ordering. Therefore reasoning with defaults requires the preference ordering, and in our case the $\ll$ ordering to be a part of the model.

In separate papers [San88a,San88b] we have extended preference logic in exactly that way. Very briefly, the preferential weakest model (PWM) entailment described there uses *aggregates* of the form $\langle \Delta, \ll \rangle$, where Δ is a set of partial interpretations, and $\ll$ is a preference ordering (a partial order) on the $\sqsubset$-maximal (usually = total) members of Δ. A formula $\mathbf{D}\alpha$ is T in an interpretation J in $\langle \Delta, \ll \rangle$ iff α is T in all $\ll$-minimal members of the completion of J in Δ. The $\mathbf{D}$ operator stands for *default,* and is used in axioms for expressing default rules as well as in the conclusions, for marking those conclusions obtained by default.

Although we have not worked out and verified the details, we hypothesize that the preference rules which defined $\ll$ in parts I and II of these papers, can be expressed using the $\mathbf{D}$ operator together with operators which test for undefinedness, and that qualification defaults can then be added to further specify the $\ll$ ordering. For example the default rule for *Controls* could be expressed as the axiom schema

$$\forall a[\mathbf{M}\neg Controls(a, c) \rightarrow \mathbf{D}\neg Controls(a, c)]$$

which reads "if it is possible that a does not control c, then by default a does not control c". The operator $\mathbf{M}$ is defined so that $\mathbf{M}\alpha$ is T iff α is T or U.

The process where partial interpretations are successively strengthened, in the inner layer, also has the following formal aspect. Consider the set F of all pairs $\langle J, J' \rangle$ of partial interpretations, where J' is obtained from J by the amount of strengthening which is necessary in order to satisfy a particular axiom. Clearly the *Enforce* operation in the decision procedure described above, is a search for the fixpoints of that relation F.

The operation F has a number of interesting properties, derivable from the assumption that all axioms are $\sqsubset$-monotone, for example

$$F(J, J') \rightarrow F(J \sqcup J'', J' \sqcup J'')$$

and of course

$$F(J, J') \rightarrow J \sqsubseteq J'$$

which means that it is a strengthening of the $\sqsubseteq$ relation itself.

In order to have a neat formulation of these properties, it is convenient to first complete the lattices so that we admit all four truth-values and all eight temporal relations in the interpretations. In this way for example the $\sqcup$ operation is always defined.

The just mentioned properties of the relation F were derived from the fact that the axioms were $\sqsubset$-monotonic. It is therefore interesting to study what happens if this requirement is relaxed, particularly because formulas containing the $\mathbf{M}$ and $\mathbf{D}$ operators mentioned above, are not in general $\sqsubset$-

monotone. An earlier paper [San85] contains results about such inference relations over a lattice of states of partial information, and the search over structures defined by the relation.

THE DECISION PROCEDURE. The major limitation of the algorithm described here is of course its finitary character. It is oriented towards applications where there are identifiable sets of objects, properties, and actions with a moderate number of members. Presumably the algorithm can not be used (not without extensive modification, at least) for applications where one has to reason about, and quantify over, very large or infinte sets of objects or actions. This finitary character however is shared not only with the usual test examples for reasoning about actions, but also with the whole current generation of knowledge-based systems technology.

If everything is finite, then the computation in the inner layer searches a finite set of partial models, and is therefore guaranteed to terminate.[6] The outer layer of the algorithm, on the other hand, searches a domain of hypotheses $\langle J, \xi \rangle$ which can be extended indefinitely by adding more actions to the P component of J. It is therefore not guaranteed to terminate.

The constrainer rules that were discussed above apply to the inner layer of the procedure. In the outer layer the need for search control is actually even more pressing, since breadth-first search according to the $<$ order will never find solutions with a non-empty abnormal change set; it will indefinitely extend P instead.

Methods for search control in the outer layer is therefore a natural topic for continued research. It would seem reasonable to use the same methods as for the inner layer, and for example introduce constrainer rules which imply the existence of an action in situations where it is necessary to introduce one more action in order to account for some known property change. The extensions that are required are however not trivial.

Similarly, it would be important to have methods for "scavenging" work which has been done in the inner layer, for re-use if the procedure has to start over in the outer layer.

An experimental implementation of the semantic decision procedure would be a very valuable tool for gaining practical experience with these methods, and particularly with the use of constrainer rules. In order to work systematically with constrainer rules, one would also need formal criteria for their efficacy, and an automatic procedure for calculating them. The method described in this paper does not serve that purpose.

In spite of these clear limitations, the logic and decision procedure described here represent a coherent method for the central problems of non-monotonic temporal reasoning, including a firm theoretical basis, and a decision procedure which is formally correct and common-sense reasonable at

[6] as long as the axioms are written in the prescribed forms.

the same time.

References

[dK86] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Gab85] Dov Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. In K.R. Apt, editor, *Logics and Models of Concurrent Systems*. Springer Verlag, 1985.

[Kle52] S. Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.

[Mak88] D. Makinson. General theory of cumulative inference. In Michael Reinfrank, editor, *Non-Monotonic Reasoning and Reason Maintenance*. Springer, 1988.

[San85] Erik Sandewall. A functional approach to non-monotonic logic. *Computational Intelligence*, 1(2):80-87, 1985.

[San88a] Erik Sandewall. An approach to non-monotonic entailment. In Z.W. Ras and L. Saitta, editors, *Methodologies for Intelligent Systems II*, pages 391-397. North-Holland, 1988.

[San88b] Erik Sandewall. The semantics of non-monotonic entailment defined using partial interpretations. Manuscript, to appear.

[Sho88] Yoav Shoham. *Reasoning about Change*. MIT Press, 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-88-29

Upplagans storlek:
Number of copies:

Datum:
Date: September 1988

Projekt:
Project:

Titel:
Title: Non-Monotonic Entailment for Reasoning about Time and Action
Part III: Decision Procedure

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. A semantics-based decision procedure for reasoning about time and action is described. Given a temporal logic where $\Gamma \models_{\ll,<} \alpha$ is defined using preference semantics, the decision procedure identifies a set of partial interpretations (or interpretation schemas) which are $\ll, <$-preferred models for all axioms in Γ. Conclusions α can then be tested by evaluating them in the computed partial models. The paper also describes how safe control rules (i.e. rules which do not change the result of the computation, but which eliminate unnecessary branches from the search) are entailed by the general (not scene-specific) axioms, using a special variant of preferential entailment.

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-88-29	**Erik Sandewall:** Non-Monotonic Entailment for Reasoning about Time and Action. Part III: Decision Procedure.
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	[illegible] Paradigm for Distributed [illegible] *OMPCON'SPRING 88*,