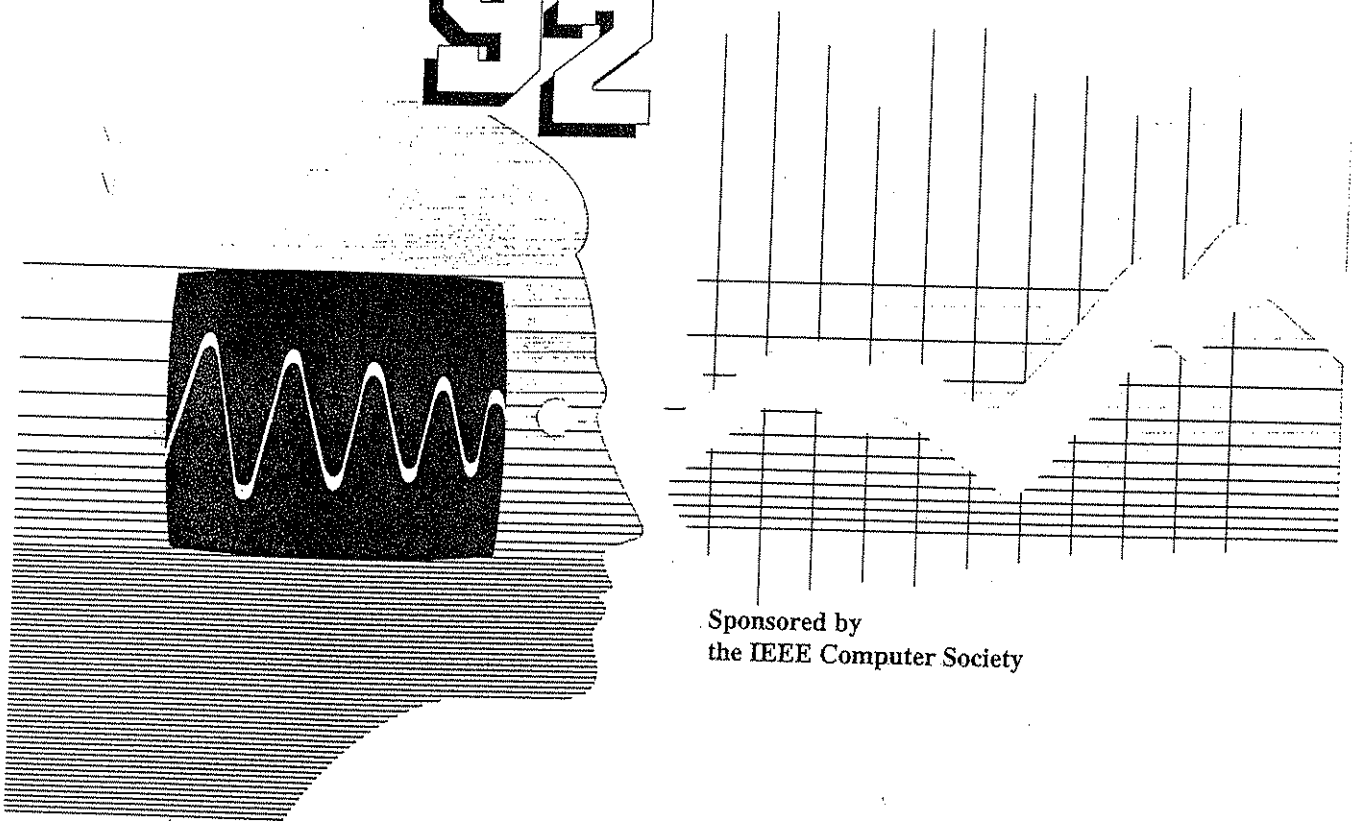
Fourth International Conference on

TOOLS WITH ARTIFICIAL INTELLIGENCE

TAI 92

November 10–13, 1992
Arlington, Virginia, USA



Sponsored by
the IEEE Computer Society

IEEE Computer Society Press

The Institute of Electrical and Electronics Engineers, Inc.

Proceedings

Fourth International Conference on

Tools with Artificial Intelligence TAI '92

November 10-13, 1992 Arlington, Virginia

Sponsored by
IEEE Computer Society

In cooperation with
SUNY — Binghamton

IEEE Computer Society Press
Los Alamitos, California
Washington • Brussels • Tokyo

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society Press, or the Institute of Electrical and Electronics Engineers, Inc.

Published by the
IEEE Computer Society Press
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Computer Society Press Order Number 2905-02
IEEE Catalog Number 92CH3203-7
ISBN 0-8186-2905-3 (paper)
ISBN 0-8186-2906-1 (microfiche)
ISBN 0-8186-2907-X (case)
ISSN Pending

Additional copies can be ordered from

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Service Center
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331

IEEE Computer Society
13, avenue de l'Aquilon
B-1200 Brussels
BELGIUM

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN

Production Editor: Edna Straub
Cover: Joe Daigle
Printed in the United States of America by Braun-Brumfield, Inc.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Welcome

On behalf of the TAI Committees, we welcome you to the Fourth International IEEE Conference on Tools with Artificial Intelligence (TAI '92). We believe that the TAI '92 meeting will be informative and beneficial to you and to your future plans.

This year's TAI meeting is based on the hard work of many individuals. I would like to thank the members of all the conference committees for their contributions to the Fourth TAI Conference. Especially I would like to thank the Program Chair, Harry Stephanou, the Vice-Chairs, and the Organizing Committee Chair, Nikolaos Bourbakis, for their dedication to making the Fourth TAI Conference successful.

I would also like to thank the invited and keynote speakers for their fruitful talks, and the staff members of the IEEE Computer Society for their support in our efforts.

General Chair, TAI '92
C.V. Ramamoorthy

This volume contains a total of 69 papers (43 regular papers, 16 concise papers, and 11 poster papers). AI topics covered include AI tools for real-time systems; production systems match; software automation; neural nets; AI for software engineering, scheduling, and planning; search; fuzzy control; learning; programming systems; database and knowledge base systems; natural language and text processing; knowledge engineering tools; matching and classification; knowledge representation and acquisition; rule-based systems; and reasoning and expert systems. These proceedings also contain position papers for four panels: "Is Production System Match Interesting?" "Knowledge Acquisition and Process Acquisition," "Nonmonotonic Logic vs. Fuzzy Logic," and "AI Tools: Who Pays the Bills?"

This volume provides an excellent set of articles on the state-of-the-art for the design, development, and evaluation of AI tools. It is also a good reference for AI methods and AI applications. We believe it to be profitable for readers in the AI field.

We wish to express many thanks to all the members of the TAI committees and the papers reviewers for their dedicated help, to the Computer Society staff members, A.M. Kelly, A. Copeland, E. Straub, M. Johnson, J. Harward, M. Kabza, and also to the students, S. Paranjpe and S.M. Mortazavi, for their support in making this volume possible.

Editors: *N. Bourbakis*
H. Stephanou
C.V. Ramamoorthy

Table of Contents

Proc. of the 1992 IEEE
Int. Conf. on Tools with AI
Arlington, VA, Nov. 1992

Applying a Time Map Manager in a Real-Time Expert System for Alarm Filtering.

Thomas Chehire

Thomson-CSF/SDC
18, Avenue du Maréchal-Juin
92366 Meudon-la-forêt. France
Tel: +33 (1) 47603985
Fax: +33 (1) 47603505
Email : Thomas.Chehire@eurokom.ie

Eva Onaindia

Universidad Politecnica
de Valencia, DSIC
Camino de Vera s/n. POBox 22012
46071 Valencia. SPAIN
Email: onaindia@xaloc.aii.upv.es

Abstract

We will address in this paper the issue of temporal reasoning in a on-line real-time expert system in the field of process control.

First we describe an alarm filtering problem, then we propose to integrate a point-based time map manager (TMM) with an inference engine, to manage the temporal constraints. The TMM manages only imprecise future time points and must comply with the constant modification of the current time and thus with the removal of time points that are moved to the past.

Temporal restrictions in rules left-hand-sides are also described, together with the needed management of pending queries that may receive an answer when the uncertainty pervading the future is reduced.

1: Introduction

Several formalisms have been proposed for temporal reasoning over the past years. Allen's interval algebra [1], Vilain and Kautz's point algebra [5], Dean and McDermott's time map [3] and Dechter, Meiri and Pearl's metric networks [4]. Temporal reasoning is usually seen as a constraint satisfaction problem, where the variables are temporal points or intervals.

In a real-time expert system, efficiency and the continual evolution of what is considered as past, current, and future facts, are important requirements that restrict the expressiveness power of the temporal constraints. An alarm filtering problem is described, and the needed temporal reasoning mechanisms are detailed. A restricted time map manager has been coupled with an OPS-like inference engine, allowing the prediction of the process evolution.

Temporal dependencies management is integrated in the TMM [7], a scheduler assists the TMM for simple temporal constraints such as facts validity.

2: Problem description

A number of process variables (observable or not) are identified in the to-be-controled plant. A trend (increasing or decreasing) in a process variable can cause different trends in other process variables. Causal links are said to be positive if an increase in the source process variable causes an increase in the destination process variable, they are said to be negative in the other case. Moreover, the influence of a process variable over another one is not instantaneous, minimum and maximum delays are thus attached to each causal link.

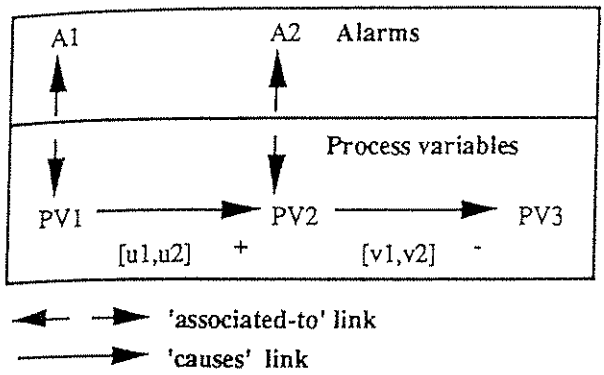
Finally alarms will fire when some process variables exceed their threshold. The known problem in process control is the overwhelming number of correlated alarms that can occur in a short period of time, confusing the process control operator.

In order to filter the correlated alarms, and thus reduce the number of alarms received by the control room operator to only the independent ones, it is proposed to predict all alarms that can occur as a result of a new incoming alarm. When new incoming alarms match the predicted alarms, it is then unnecessary to present it to the control room operator. Instead one should check if the previous alarm, that caused this new alarm to fire, has been handled correctly and that the process will soon return to normal operation mode.

Let us illustrate this through the following example:

- Alarms A1 and A2 are associated to process variables PV1 and PV2,
- an increase in process variable PV1 will cause an increase in process variable PV2, after u1 seconds and before u2 seconds.
- an increase in process variable PV2 will cause a decrease in process variable PV3, after v1 seconds and before v2 seconds.

These are recorded on a blackboard as shown in the following figure:



Each node on the blackboard has static and temporal slots. Temporal slots can record some past values, and predictions can be made about possible future values.

Past values have precise begin and end times (time interval during which the slot value remained unchanged). The current value has a precise begin time and an unknown or constrained end time.

Both begin and end times of predictions are imprecise and may be constrained.

Current values are created with put-slot-value, predictions with predict-slot-value:

```
(predict-slot-value  <blackboard
                                node>
   (<slot> <value>)
   (begin  <temporal constraints
             for the begin time>)
 [ (validity  <maximum distance
            between end and begin>)
  |(end  <temporal constraints
             for the end time>)])

(put-slot-value  <blackboard node>
   (<slot> <value>)
 [ (validity  <maximum distance
            between end and begin>)
  |(end  <temporal constraints
             for the end time>)])
```

These of course can be used as a user language to modify the blackboard, but can also be used in rules right-hand-sides.

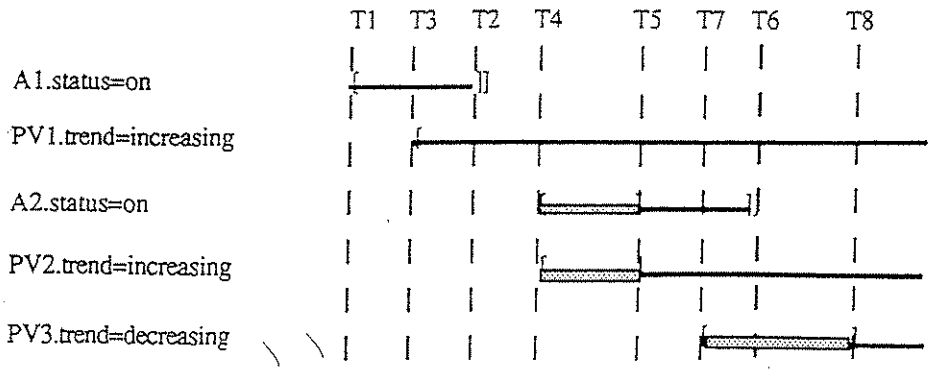
Let us illustrate this with the previous example:

When an alarm is on, then the associated process variable is increasing, going above a security threshold (or decreasing going under a security threshold). A rule encoding such knowledge would match on the modification of the 'status' slot of an alarm to the value 'on', and would use in its righ-hand-side a 'put-slot-value' to update the trend of the associated variable. The begin time of the trend will be the time of the rule firing, since it is not allowed to modify the past. This means that if the trend of the variable was previously unknown and the alarm becomes active, then the begin time of the trend modification is not the real-time at which the trend was really increasing (for example), but it is the time when it is necessary to take it into account since an alarm was fired. On the other hand, the end time of the trend modification is unrelated to the end time of the activated alarm. An alarm will be considered by a control operator within 3 seconds of time, while the cause of the alarm must still be handled.

Finally to propagate the trend modification of a process variable to other related process variables, it is necessary to write a rule whose left-hand-side matches on the trend modification of the first variable and uses a 'predict-slot-value' in its right-hand-side to relate the begin time of the trend modification of the second variable to the begin time of the trend modification of the first variable.

Let us consider the situation depicted in the following figure:

- at time T1, the A1 alarm is active the slot 'status' is set to 'on' for a maximum of 3 seconds (it will have to be unbound before T2=T1+3 seconds.



- the rule responsible for updating the trend of the associated variable is fired at time T3, and sets the trend of PV1 to 'increasing' with an unknown (possibly infinite) end time.

- the rule responsible for propagating the trends along the causal links, fires and predicts the increase of variable PV2 and the potential firing of alarm A2. The begin time is the same for both predictions and is within the interval [T4 , T5]; T4=T1+u1, T5=T1+u2. The end time T6 for the alarm A2 is at most 3 seconds after its begin time, T6=T5+3 seconds.

- the propagation rule fires again and predicts the decrease of variable PV3, its begin time is related to the begin time of PV2 and is within the interval: [T7, T8]; T7=T4+v1 T8=T5+v2.

3: Using a TMM to manage the temporal constraints:

Since current values and predictions for blackboard nodes can have imprecise begin and end times, we selected a point-based TMM.

The end time for a current slot value is represented by a Time Map Node (TMN). Predictions have time map nodes for both their begin and end time.

The blackboard node will record for each current or predicted slot value its associated TMNs, each TMN can be linked to other TMNs through temporal constraints.

The simplest constraint between two TMNs is a minimum and a maximum distance.

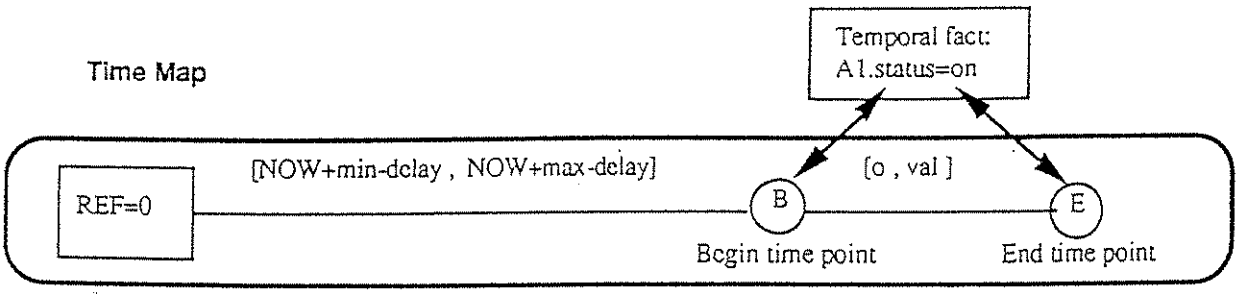
Each TMN knows if it is associated to a begin or end time of a prediction or to the end time of a current value. Let us illustrate this with the resulting time map (bottom figure) after a call to:

```
(predict-slot-value  A1
    (status on)
    (begin (and (after NOW min-delay)
                (before NOW max-delay))
    (validity val))
```

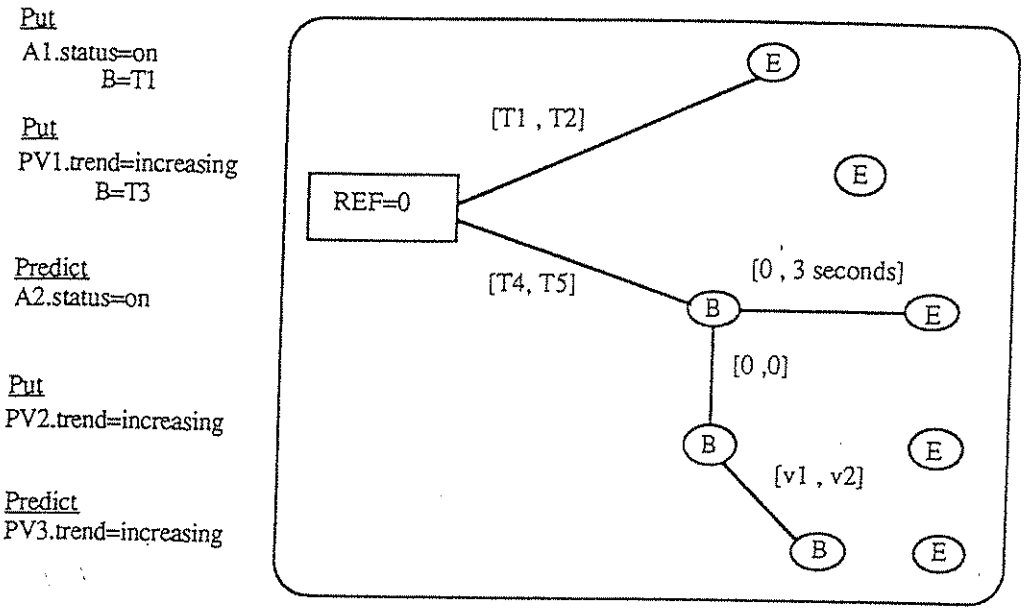
NOW represents the current time.

We provide the body of the two rules encoding the prediction of alarm firings, the evolution of the time map is then illustrated in the following page.

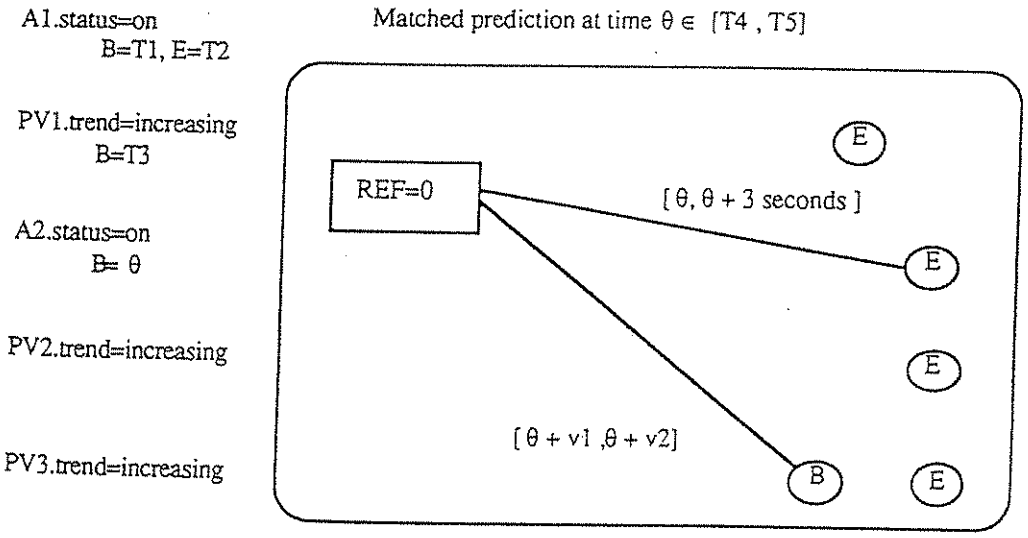
```
(rule  from-alarm-to-process-variable
;;; When an alarm fires, deduce the trend
;;; of the associated process variable.
    (bb_node  ?alarm
        (bb_level  alarm)
        (holds  (status on))
        (associated-to (?proc-var  ?trend)))
  ->
    (put-slot-value  ?proc-var
                     (trend  ?trend)))

(rule  from-pv-to-alarm-or-pv
;;; Follow the causal network to predict
;;; all possible process variables trends
;;; and potential alarm firings.
 (bb_node  ?pv1
  (bb_level  process-variable)
  (holds  (trend  ?trend)  (?t1,?t2))
  (causes  (?pv2  ?positive  ?dmin  ?dmax)))
 ->
 (bind  ?alarm
      (get-slot-value  ?pv2  associated-to))
 (cond  ((null ?alarm)
         ;; The ?pv2 process variable is not
         ;; associated to an alarm.
         (bind  ?induced-trend
              (if  ?positive
                   ?trend
                   (inverse-trend  ?trend)))
        (predict-slot-value  ?pv2
          (trend  ?induced-trend)
          (begin  (and  (after  ?t1  ?dmin)
                        (before  ?t1  ?dmax)))))
       (T  ;; Predict the alarm firing
           ;; and let the process variable
           ;; trend be linked to the alarm
           ;; firing through the first rule
         (predict-slot-value  ?alarm
           (status  on)
           (begin
            (and  (after  ?t1  ?dmin)
                  (before  ?t1  ?dmax)))))))
```



Time map layout at a time between T3 an T2, after all rules have been fired:



Time map between T4 and T5, if alarm A2 fires at θ:



4: Blackboard and TMM nodes, semantics of predict-slot-value and put-slot-value:

Each blackboard node has static slots set at initialisation time and that cannot be modified, together with temporal slots that can have current and past slot values with eventual predictions of future values.

Each value for a temporal slot is stored in a temporal fact which records the begin and end times of the value.

The Time Map needs only to manage the imprecise or unknown future. The begin time of current slot values, together with begin and end times of past values are precise dates. Only the end time of current values and begin and end times of predictions will have to be represented in the time map.

The begin time of a prediction can be constrained to be within a certain absolute time interval ([minimum begin time, maximum begin time]), or a relative temporal distance from another time map node.

The end time of a temporal fact can be constrained in the same manner as a begin time. When a slot value has a maximum validity, the end time is just related to its begin time with the [0, validity] temporal constraint.

4.1: Using a scheduler to manage absolute time constraints:

When a begin or end time point is not related to another time point but is constrained by an absolute time interval, this means that the upper bound of the interval is the time at which the time point should be retracted from the time map, since it would violate the temporal constraints otherwise.

If the time point is a begin time of a prediction, then the prediction is not fulfilled and it should itself be retracted. If it is the end time of a temporal fact, then it must be the end time of a current value since the end time of a prediction must be after its begin time. In this case, the current value is not valid any more and must be retracted.

In order to implement such a functionality, a scheduler is used to execute actions at a given absolute time in the future. As soon as a time map node is related to the time reference (minimum and maximum absolute time interval), the action of retracting this node from the time map is scheduled at the upper bound of its associated temporal window. If a current slot value is changed before its maximum end time, then the 'retract-end-time-point' action is unscheduled.

When a new value is given to a slot, predictions are first checked to try and find one with the same slot value and compliant begin time window. In this case, the prediction is transformed into a current value, its begin time point is now a precise time and the associated time map node has to be retracted from the time map, the 'retract-begin-time-point' action is unscheduled.

4.2: Propagations in the time map network:

The time map manages only future dates, some nodes will thus disappear while the current time advances. Two situations have been particularly noted: replacing a node by a precise time (a matched prediction has now a precise begin time, or a current value is modified or has reached its maximum validity), and removing a node (a prediction was not fulfilled).

If a node is replaced by a precise time all related nodes must be updated, while if it is removed all related nodes must also be removed.

4.3: Semantics of predict and put slot value

Put-slot-value modifies the current value of a slot, its begin time is set to the current time, and its end point is usually a time map node related to the current time by a maximum distance (the temporal validity of the value).

Predict-slot-value creates predictions whose begin point may be in some absolute time interval in the future, or related to other time map nodes.

A call to put-slot-value will first check if the new value was already predicted, and in this case update the begin and end times of the prediction instead of creating a new temporal fact.

The advantage of course is that rules can fire on predictions, and complete lines of reasoning can depend on them. When predictions are matched, the conclusions depending on them are confirmed instead of firing the rules on a new current value. This allows to run much faster on slot modifications, if time was available in the past to draw all possible inferences from the predictions.

One special situation occurs when a rule fires on predictions and uses a put-slot-value in its righ-hand-side. In this case the new value is not considered as a current value, but as a future value depending on the predictions that matched the rule's left-hand side. The begin point of this special future temporal fact is set to be after each of the begin points of the involved predictions. Only when all left-hand side predictions are fulfilled (matched by current values) will the special temporal fact become a current value. While a normal prediction created by predict-slot-value in the rule's right-hand side would still remain a prediction and will have to match a new put-slot-value to become current.

As seen above, it is sometimes necessary to constrain a time map node to be the earliest or latest amongst a group of time map nodes. Two special kinds of nodes have been introduced to implement such constraints: MAX and MIN. These nodes link many input nodes to one output node. When all but one input nodes of a MAX node have precise times, the remaining input node is directly linked to the output

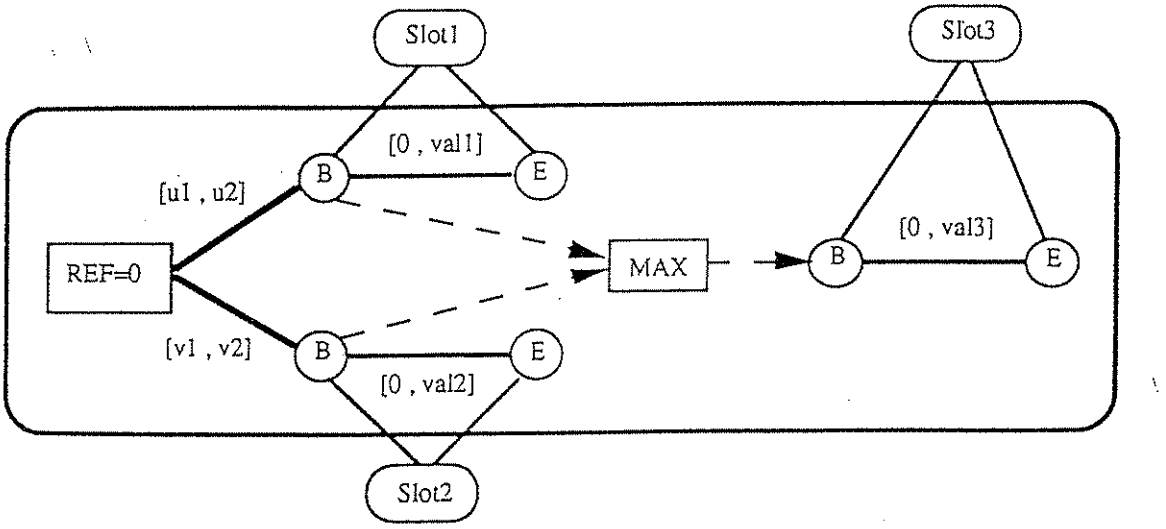
node ensuring that they will have the same precise time in the future. On the other hand, as soon as one input node of a MIN node has a precise time, its output node receives the same precise time. These two behaviours depend on the hypothesis that time is flying forward (the clock is increasing) and that it is not possible to modify the past.

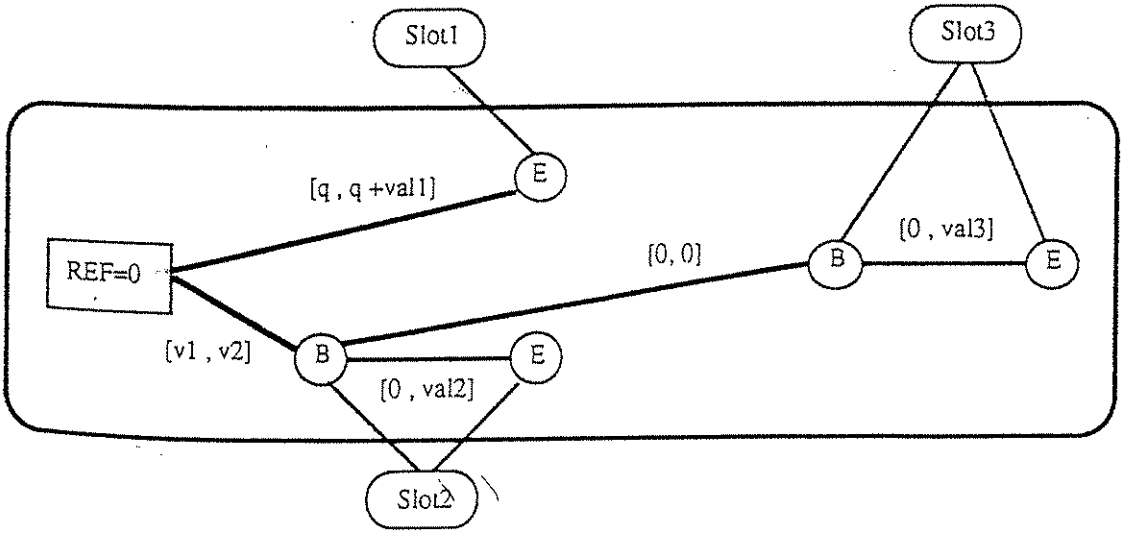
Let us illustrate the behaviour of put-slot-value with a simple example involving three slots of a blackboard node:

The first figure shows the time map and temporal facts just after firing the "test" rule.

```
(rule  test
   (bb_node  ?o
      (holds  (slot1  ?v1))
      (holds  (slot2  ?v2)))
   ...
 ->
    ...
   (put-slot-value  ?o  (slot3  ?v3))
   )
```

The second figure shows the time map just after matching the prediction for slot1 with a new value at time θ. The MAX node is removed and replaced by a simple temporal constraint:





5: Temporal tests

The rule language has been extended to include temporal tests on the relative distance of time points. Future time points are imprecise and it is not always possible to provide a yes/no answer to a temporal query. The integration of a TMM in an on-line expert system brings two particular concerns:

- the current-time is always increasing, and imprecise time points in the future will eventually receive a precise date and thus be moved to a known past,
- an "unknown" answer to a temporal query delays a rule firing until time map updatings can eventually result in a yes/no answer.

The peculiar aspect of our TMM is thus that it manages only time points that are in the future with regard to any current real time, and that it must maintain pending queries in order to provide delayed yes/no answers that can be exploited by an inference engine. We will illustrate these two aspects in the following.

The language for temporal queries is:

```
simple-test ::=
  (NOW  before | after  <date>
                        [ + | -  <duration>| ]
| (<date>  before | after [ at  <date>
                        [ + | -  <duration>| ]

temporal query ::=
  <simple-test>
| (NOT  <temporal query>)
| (AND  <temporal query>  <temporal query>+)
| (OR   <temporal query>  <temporal query>+)


date      ::=  <time point>
   | <a number of time units: absolute date>
duration ::=  <a number of time units>
NOW       ::= represents the current time

Example:

(rule test
  (bb_node tank-1
   (holds  (pressure  low)  (?t1, ?t2])
   (holds  (temperature  high)  (?t3, ?t4]))
  (temporal_test
        (?t1  before  ?t3  -  (minutes 10)))
  ->
   ...)
```

The simplest queries are thus to know if one date is before or after another one, four cases can be identified for comparing the current time and an absolute date, the current time and a time point, an absolute date and a time point, or two time points.

5.1: Comparing the current time and an absolute date:

A given date can be in the past (before the current time) and will remain in the past, or it can be in the future until the current time reaches this date. This last case "(NOW before <precise time in the future>)" must thus be handled by the scheduler described previously, the answer to the query is 'yes' and the scheduler should wake up the TMM at the given precise time to send a new answer of 'no'.

5.2: Comparing the current time and a time point:

A time point represents an imprecise future, it is thus after the current time until it receives a precise date and is moved to the past. The answer to the query "(NOW before <time point>)" is thus 'yes', but an action is attached to the time point so that when it receives a precise date the query is handled as in the previous case.

5.3: Comparing an absolute date and a time point:

A time point can be related to other time points by a temporal constraint (minimal and maximal distance). However, in our restricted time map, a time point can have many successors but just one predecessor. Predecessors of the given time point are searched to look for one ancestor with an absolute temporal window (instead of a relative temporal window to its predecessor), if such an ancestor is found, an absolute temporal window can thus be computed for the time point involved in the query. Let [min ,max] be the temporal window for the time point, if the given absolute date is lower to min or greater to max, the query will have a yes/no answer. Otherwise it is necessary to maintain a pending query on that time point, so that whenever its uncertainty is reduced the query is tried again. For that purpose, the time point and all its predecessors are marked so that whenever one has a new absolute temporal window, the absolute temporal window of all its marked successors are computed, up to the time point with the pending query.

5.4: Comparing two time points:

When the time points have a common ancestor, it can be the time reference, in which case both time points have absolute temporal windows, or it can be a time point, and in that case, both points have relative temporal windows to this common ancestor. If the relative or absolute temporal windows do not intersect, then the query has a yes/no answer, otherwise we must mark all predecessors back to but except the common ancestor so that whenever one of these ancestors has a new absolute temporal window, a propagation is made to all marked successors up to the time points with the pending query.

In the case when the two time points have no common ancestor, then all ancestors are marked.

6: Conclusion

A restricted time map manager has been integrated in a blackboard based architecture with an OPS-like inference engine. The TMM manages predictions about slots values of blackboard nodes, temporal restrictions can be used in rules left-hand-sides. In a real-time expert system, this allows to reason about possible future facts when the load of the system is not too high. When predictions are matched all deductions made on the basis of the matched predictions are confirmed instead of having to fire the rules. The architecture has been validated on an alarm filtering problem, the incoming alarms are first compared to predicted ones and in this case their handling is much faster than in the case of unpredicted ones. Alarm prediction is however a low priority task, and is undertaken only when time permits it.

The restricted TMM is efficient since no contradictions can appear, the time map is a set of independent trees. Allowing graphs with disjunctions and contradiction management, while integrating the notion of current time and pending queries is a possible extension to the current architecture, though it may be unfeasible in a real-time context.

Acknowledgements

The work described in this paper has been funded by the ESPRIT II REAKT project on real-time expert systems. The partners involved in REAKT are: Thomson-CSF/RCC (prime), SYSECA, CRIN-INRIA, ETNOTEAM, GEC-MARCONI, Grupo Mecanica del Vuelo, Universidad Politecnica de Valencia, and Computas expert systems.

We would like to thank Federico Barber, Vicente Botti, and Alfons Crespo [2] from Universidad Politecnica de Valencia, for the background knowledge they provided on time map management.

The use of a time map manager is under consideration at the air traffic control division of Thomson-CSF/SDC.

References

[1] Allen J.F. Maintaining knowledge about temporal intervals. CACM, 26(11) p. 832-843 (1983)

[2] F.barber, V. Botti, A.Crespo. Temporal expert systems. IASTED'90 Los Angeles.

[3] Dean T.L., McDermott D.V.
Temporal Data Base Management.
Artificial Intelligence, 32 p. 1-55 (1987)

[4] Dechter R., Meiri I., Pearl J.
Temporal constraint networks.
Artificial Intelligence, 49 p. 271-277 (1991)

[5] Vilain M., Kautz H. Constraint propagation algorithms for temporal reasoning.
Proceedings of AAAI-86 p. 377-382 (1986)

[6] Meiri I. Combining qualitative and quantitative constraints in temporal reasoning.
Proceedings of AAAI-91

[7] Honeywell Systems & Research Center
ß-TMM Functional Description (1991)

[8] Ghallab M., Mounir Alaoui A.
Managing efficiently temporal relations through indexed spanning trees. Proceedings of the 11th IJCAI, p. 1297-1303 (1989)