An Extended Functional Representation in Temporal Planning: towards Continuous Change

Romain Trinquart and Malik Ghallab

LAAS-CNRS, 7 avenue du colonel Roche, 31077 Toulouse, France,
romain.trinquart@laas.fr, malik@laas.fr

Abstract. This paper is concerned with temporal planning relying on CSP-based functional representations. These powerful representations are today mostly restricted to the use of piecewise constant functions ranging over finite domains.
We are proposing here an extension that brings a significant enhancement in the expressiveness of the representation, towards handling continuous change. This extension consists mainly in allowing *piecewise linear functions over continuous domains*. We have studied this extension and we are currently implementing it in the context of the IXTET planner. However this extended representation is not specific to that planner and it can be useful to most temporal planners. We show how IXTET syntax, planning algorithm and control can be simply extended to this class of functions. We then consider the more significant modifications required in the two constraints managers for handling temporal and atemporal CSPs.

1 Introduction

Planning is about time; it is mainly the synthesis of a temporal projection over the future for achieving some desired goal. Time appears to be the main component in a planning ontology. However, *classical planning* has abstracted away time in state transition systems. This restrictive assumption proved to be fruitful in developing planning algorithms and techniques. But its well-known drawbacks in expressiveness - concurrency, duration, persistence, actions for maintaining values, temporally qualified goals and dynamics - and the lack of direct applications of this technology argue more and more for pursuing the development of explicit temporal planning.

Instead of global, still pictures of the entire world, i.e. states, and their transitions, the temporal representation we are interested in here focuses on local individual evolutions of state variables. It is a *functional representation*: each state variable is a partially specified function of time, called a *timeline* or *history*. It is a *CSP-based representation*: these temporal functions are linked through a set of temporal and atemporal constraints into a globally consistent temporal projection.

This potentially powerful representation has been developed and used by several planners, among which IXTET [Laborie 95], RAX-PS
[Muscettola 97] or parcPlan [El-Kholy 96]. But is has been mostly restricted to piecewise constant functions ranging over finite domains. The contribution of this paper is to extend the representation to the use of piecewise linear functions, ranging over continuous or discrete domains.

This representation requires significant extension in the CSP manager of atemporal variables, going from finite domains to mixed finite and continuous domains. It further introduces an interesting coupling between atemporal and temporal variables that were up to now managed separately. This coupling requires another extension of the Time-Map manager.

We studied these extensions and we are currently implementing them in the context of the IxTeT planner. The paper presents the proposed extension within IxTeT notations and construct. However the approach is not specific to that planner, it can be useful to most temporal planners. To be self-contained (and to further promote the CSP-based functional representation) we will describe it in section 2 together with the proposed extension. We will develop in section 3 the required extensions in planning algorithms and control. We then focus on the two atemporal and temporal CSP networks, their extensions and links required by the new type of functions introduced. Algorithms will be described and discussed, however no empirical results on the performance of the extended planner are yet available. Similarly, we'll focus the paper on state variables without considering at this stage the resource handling capabilities of IxTeT. We conclude on the expressiveness of the extended representation and its relationship to other approaches.

2 Representation

In IxTeT, a dynamic domain is described by a set of temporally qualified attributes (multi-valued fluents), each being a k-ary mapping from a finite domain into a finite range. Time is considered as a linearly ordered set of instants. We use time-points as the elementary primitive. Conjunctions of constraints (both symbolic constraints of the time-point algebra and numerical ones) can be expressed between time-points. As will be discussed later, the system does not handle disjunctions. Thus numerical constraints are upper and lower bounds on the temporal distance between two time points.

To describe the dynamic of the world, a propositional reified logic formalism is used, where attributes are temporally qualified by two predicates : *hold* and *event*. The persistence of the value v of an attribute p during the interval *[t,t'[* is expressed by the assertion *hold(p:v,(t,t'))*. The instantaneous change of the value of p from v to v' at time t is expressed as *event(p:(v,v'),t)*.

A partial plan, as well as a planning operator (called a *task*), is represented by a *chronicle*, that is to say a conjunction of temporal predicates (*event* and *hold* predicates) and of constraints on time-points and on atemporal variables used as values or arguments of attributes. The constraints allowed in IxTeT are binary constraints of the form $t \leq t'$ or $t - t' \in [l, u]$ (where t and t' stand for time points) and $x = y$, $x \neq y$, $x \in D$ or $x \in D_x \Rightarrow y \in D_y$ (where x and y stand for atemporal variables).

Chronicles are the core of the representation from the planner's point of view. This representation offers considerable expressiveness by allowing to distribute events and persistence conditions within a task at different time points. Through chronicles, we benefit from a partial specification of the world along some threads caused by actions. Note that an *event* expresses, within a single proposition, both a precondition and an effect of an action : *event(p:(v,v'),t)* requires that p has value v before t and causes p to be equal to v' after t; furthermore this can be expressed anywhere with respect to the start and end points of the action.

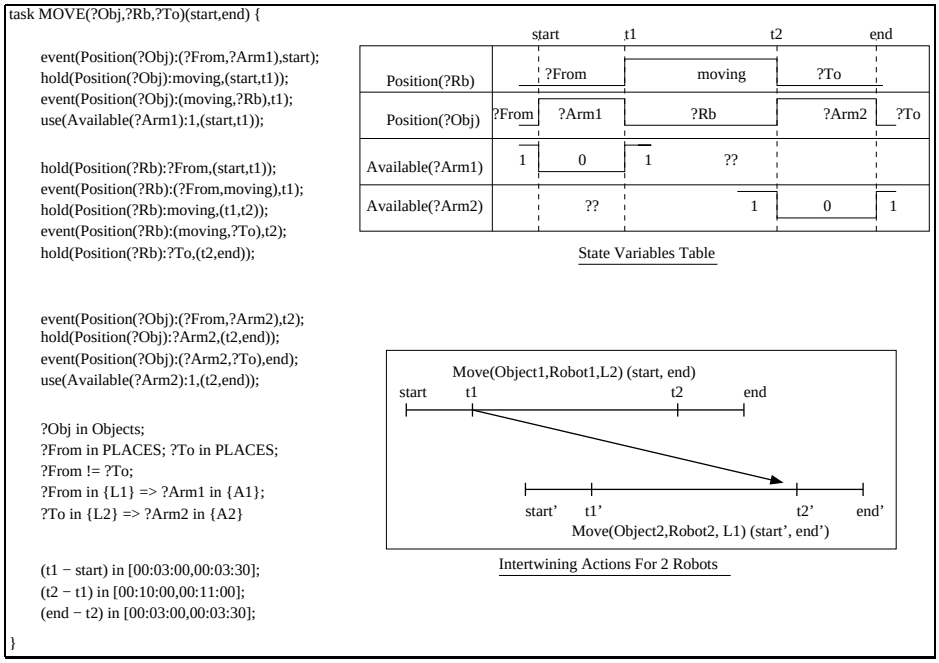


Table 1. A Task Example in the IXTET Formalism

Table 1 presents an example of a task in a simple domain, where a robot can move an object from one place to another. It shows how two instances of this task can be interleaved for two coordinated robots. This instance of a chronicle shows several key points in the representation, such as events occurring at different time points and the possibility for the planner to compute constraints that lead to interleaved actions, thus providing more efficient and flexible plans. We should also point out that tasks variables within a chronicle are not necessarily instantiated but only constrained.

Proposed extension Clearly, a conjunction of *event* and *hold* predicates on some attribute p corresponds to a partially specified function of time ranging over the finite domain of p. An *event* is a step of that function at some time point. A *hold* is a constant value over some time interval. Only piecewise constant function ranging over finite sets can be expressed with *event* and *hold* predicates qualifying attributes over finite domains.

Very few dynamic domains are easily approximated with piecewise constant functions. The position of a robot or the orientation of a satellite do not change along a single step. Admittedly, intermediate positions or orientations are not always of interest at the planning abstraction level. The solution here is to use an unspecified attribute value, e.g. *moving* in the above example, between the values of interest. However, this makes a poor use of the flexibility of the CSP-based functional representation, in particular for interleaving activities and adding incrementally specifications and constraints on attributes while planning. The

moving value cannot be combined with values of other attributes to permit or to forbid other activities at some intermediate positions or orientations. In order to do that, one may break down the change of the corresponding attribute into several steps. However this approach is not very satisfactory; it complicates the specification of planning operators. Furthermore, it can be less efficient than handling directly an extended representation.

We propose here to add to *event* and *hold* a third temporal predicate, called *change*. The expression $change(p : (v, v'), (t, t'))$ specifies that the value of attribute p changes linearly from v to v' during the interval t to t'. Predicate *change*, together with *event* and *hold*, extends the representation to partially specified piecewise linear functions.

The specification of the evolution of p over $[t, t'[$ will be completed by an explicit constraint of the form $v' - v \in D$ or $v' - v = a * (t' - t)$. This last kind of constraint brings about an interesting aspect of the proposed extension since it enables one to connect the effect of an action to its duration which is necessary to express many real world domains.

Having added predicate *change*, the assumption that attributes range over finite domains cannot hold anymore. Consequently, we also added numerical attributes ranging over intervals in the set of real numbers. Constraints concerning numeric variables are : $x \in D$ to specify an initial domain, $x \in D_x \Rightarrow y \in D_y$ to express range dependency (these constraints might mix discrete and numeric variables) and also the two kind of constraints on difference mentioned above.

These two extensions require some slight generalization of the planning and control algorithms, but they mainly require a re-design of the atemporal CSP manager, and, because of the coupling introduced, of the temporal constraint manager as well.

3 Planning Algorithm and control

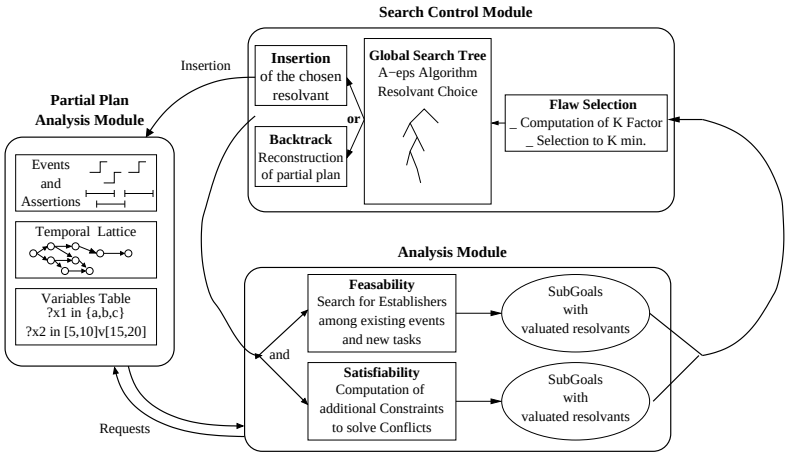


Fig. 1. The Different Modules in IxTeT

IxTeT performs a least commitment search in a space of partial plans [Weld 94]. Nodes of the search space are chronicles. Edges correspond to plan refinements such as adding a task or a constraint. A chronicle Π is a valid solution plan iff:

- (1) For every assertion $hold(Att(x_1,...,x_k) : v,(t,t'))$ and for every event $event(Att(x_1,...,x_k) : (v,v'),t)$ Π contains :
  - an establisher $E = event(Att(x_{est1},...,x_{estk}) : (w,v_{est}),t_{est})$ such that $(x_1 = x_{est1}) \wedge \ ... \ \wedge (x_k = x_{estk}) \wedge (v_{est} = v) \wedge (t_{est} < t)$
  - a causal link $hold(Att(x_1,...,x_k) : v,(t_{est},t))$.
- (2a) For every couple (E,H) where $H = hold(Att(x_1,...,x_k) : v_1,(t_1,t'_1))$ and $E = event(Att(y_1,...,y_k) : (v_2,v'_2),t_2)$ Π contains one of the following constraints : $(t_2 < t_1) \vee (t'_1 < t_2) \vee (x_1 \neq y_1) \vee \ ... \ \vee (x_k \neq y_k)$
- (2b) For every couple (H_1,H_2) where $H_1 = hold(Att(x_1,...,x_k) : v_1,(t_1,t'_1))$ and $H_2 = hold(Att(y_1,...,y_k) : v_2,(t_2,t'_2))$ Π contains one of the following constraints : $(t'_2 < t_1) \vee (t'_1 < t_2) \vee (x_1 \neq y_1) \vee \ ... \ \vee (x_k \neq y_k)$
- (3) The temporal constraints and atemporal constraints are consistent.

This criteria enables us to decide why and how a chronicle should be refined in order to achieve every subgoal (unexplained propositions) through the insertion of tasks and/or causal links, and in order to solve any potential conflict through the insertion of temporal or atemporal constraints.

The first two conditions above are used to detect flaws and to define resolvers. The selection of the next flaw to consider and the non-deterministic choice of a resolver are the main control decisions of the planner. The third condition on the consistency of the constraints does not direct the search process. At each step of the search, constraints are dynamically inserted. They are locally propagated through the corresponding constraint network. If the local propagation detects an inconsistency, then the search process has to backtrack. The local propagation being incomplete for atemporal variables, it is completed by a global consistency checking once a plan is found.

We are going to use the same approach for the extended representation. As an *event*, a *change* is both a precondition and an effect. Hence it may introduce a subgoal requiring an establisher. It can also produce such an establisher. Similarly there can be threats between a *change* and a *hold*, or a between *change* and an *event*, or with another *change*.

Let us consider an unexplained *change* predicate: $change(Att(x_1,...,x_n) : (v_i,v_f),(t_i,t_f))$. It can be established by another such predicate, already in the current chronicle or to be added through some task: $change(Att(x'_1,...,x'_n) : (v'_i,v'_f),(t'_i,t'_f))$ if the following conditions are consistent with the current chronicle:

$x_1 = x'_1 \ and \ ... \ and \ x_n = x'_n \ and \ v'_f = v_i and \ t'_f \leq t_i$.

In that case, the corresponding resolver is the above conjunction together with the following causal link : $hold(Att(x_1,...x_n) : v_i,(t'_f,t_i))$.

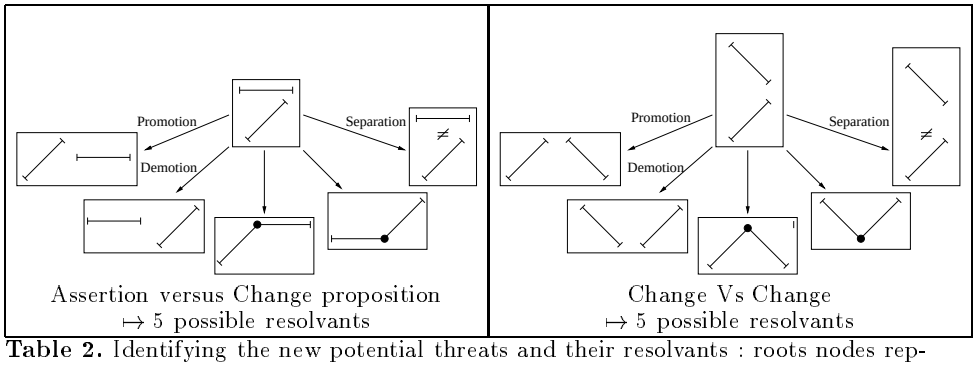
A threat between two predicates: $hold(Att(x_1,...x_n) : v,(t_1,t_2))$ and $change(Att(x'_1,...,x'_n) : (v'_1,v'_2),(t'_1,t'_2))$ can be solved by this disjunction of constraints, when consistent with the current chronicle:

$$\begin{array}{rl} (x_1 \neq x'_1) \ or \ ... \ or \ (x_n \neq x'_n) & \text{(separation)} \\ or \ (t_2 < t'_1) & \text{(promotion)} \\ or \ (t'_2 < t_1) & \text{(demotion)} \\ or \ (t_1 = t'_2 \ and \ v = v'_2) & \\ or \ (t'_1 = t_2 \ and \ v = v'_1) & \end{array}$$

Similarly a threat between $change(Att(x_1, ..., x_n) : (v_1, v_2), (t_1, t_2))$ and $change(Att(x'_1, ..., x'_n) : (v'_1, v'_2), (t'_1, t'_2))$ is solved by this disjunction of constraints:

$$\begin{array}{rl} (x_1 \neq^? x'_1) \ or \ \ldots \ or \ (x_n \neq x'_n) & \text{(separation)} \\ or \ (t_2 < t'_1) & \text{(promotion)} \\ or \ (t'_2 < t_1) & \text{(demotion)} \\ or \ (t_1 = t'_2 \ and \ v_1 = v'_2) & \\ or \ (t'_1 = t_2 \ and \ v_2 = v'_1) & \end{array}$$

The various cases of threats introduced by the extended representation are summarized in the table 2. They can all be handled in a similar way by adding to the current chronicle a conjunction of temporal and atemporal constraints. The remaining problem is how to handle these extended constraints.



Assertion versus Change proposition
$\mapsto$ 5 possible resolvants

Change Vs Change
$\mapsto$ 5 possible resolvants

Table 2. Identifying the new potential threats and their resolvants : roots nodes represent potential conflicts, branches correspond to possible resolvers.

4 The Constraints Managers

IxTeT has three different specialized constraints managers : one dealing with atemporal variables, one dealing with time points and a third one dealing with resources. In order to cope with the new kind of constraints needed to handle the extension to continuous change, we need to enhance the atemporal constraints manager and to bring about closer interactions between this manager and the Time-Map.

4.1 The Atemporal Constraints Manager

This system handles the atemporal variables which are dynamically introduced in the partial plan during the search process, as well as the related constraints which might be introduced to solve flaws. It is actually a CSP Solver dealing with variables over finite domains.

The manager is queried by the planning system for testing if two variables can be unified or differentiated, for inserting new constraints and for checking

global consistency. The desired level of expressivity required by the planner is quite rich. We have to handle the following constraints : $x = y$, $x \neq y$, $x \in D$, $x \in D_x \Rightarrow y \in D_y$.

Thus the manager problem is that of a general binary CSP over finite domains whose consistency checking is a NP-complete problem. Because of the large number of queries from the planner to the manager, IxTeT maintains an incomplete arc-consistency checking while planning. Complete consistency checking is postponed until a plan without flaws is found.

The main drawback of this method is that it leads to "blind" backtrack : the search process might choose to develop a refinement branch that is doomed to fail because of an undetected inconsistency, thus loosing time. Furthermore, once the inconsistency is detected the backtrack point cannot be easily identified. However this criticism should be counterbalanced by the following observation : because of the least commitment strategy, carrying on the search process will lead to a reduction of the problem size by introducing equality constraints corresponding to postponed establishment.

Dealing with continuous domains : To handle *change* predicates, the planner needs to get information concerning variables whose domains are continuous. The constraints that might be posted on two numeric variables are : $x \in D$, $x - y \in D'$ and $x \in D_x \Rightarrow y \in D_y$.

This last class of constraints leads us to consider domains that are disjunctions of intervals in $\mathbb{R}$: given a constraint "$x \in D_x \Rightarrow y \in D_y$", if $Dom(y) \cap D_y = \emptyset$ then the propagation yields the insertion of the constraint $x \notin D_x$ which might create "holes" in the domain associated to x.

The local arc consistency algorithm can be adapted without too much trouble. We have to provide operators on the intervals disjunctions similar to those used for discrete set and also to define propagation rules concerning the new "difference" constraints.

Problems arise concerning the complete propagation algorithm. This algorithm performs an extensive instantiation of the variables to detect the values that will necessary lead to inconsistency. It is a forward checking search . This cannot be directly transposed to numeric variables : it makes no sense to check for all the values in a continuous domain. A naive technique consists in replacing variable instantiations by domain splitting. A dichotomy splitting can be iterated until a fixed minimal interval length is reached. We propose to use an analog method, but instead of a systematic dichotomy that does not take the constraint network into account and thus might lead to unnecessary computations, we will use the constraints to guide the splitting of intervals.

Our purpose is to have a common framework to handle both discrete and numeric variables. To achieve this goal we have got to change the notion of instantiation concerning numeric variables so as to obtain equivalent complexity. Let us point out that , in the discrete case, instantiations are performed in order to reduce the uncertainty on the value of a variable, thus enabling the solver to actually fire every constraint. But, hopefully, the triggering of the constraints concerning numeric variables does not require the variables to be fully instantiated. Among the constraints on numeric variables, only the dependency constraints propagation might be postponed because of a lack of information. The equality and domain restriction constraints will be immediately propagated

at insertion, respectively through the fusion of the equivalence class of the affected variables and through the effective reduction of the valuation domain. The distance constraints ($x - y \in D$) will be propagated by the arc consistency propagation. But to propagate a dependency constraint "$x \in D_x \Rightarrow y \in D_y$", the solver needs to know whether x (resp. y) takes its value in D_x (resp. D_y) or not. This is the kind of constraints that might stop propagation along paths and thus prevent the detection of inconsistency. The complete propagation algorithm should reduce the uncertainty on the variables enough to decide how to propagate the constraint, i.e. it should decide whether $x \in D_x$ or not. This is the key to our algorithm.

The domain of a numeric variable x will be decomposed into disjoint intervals I_k such that for every constraint "$x \in D_x \Rightarrow y \in D_y$" or "$z \in D_z \Rightarrow x \in D_x$", the following proposition is true : $(I_k \subset D_x) or (I_k \cap D_x = \emptyset)$. The complete propagation procedure will then successively try to reduce the valuation domain of x to these different intervals (I_k). For each I_k it will be able to decide how the constraints should be applied. Thus the problem of instantiating a numeric variable has been transformed into choosing subintervals in a finite set.

Since this method can also be used for discrete variables (domains will then be decomposed into disjoint subsets), it provides a simple algorithm to deal with a heterogeneous CSP containing both discrete and numeric variables for the type of constraints in a temporal planner. Table 3 presents the algorithm for complete propagation in such a CSP. The procedure Complete_Propagation computes minimal domains in the CSP. For every variable, it builds a partition of its domain according to the constraints. Then it successively shrinks the domain to every element of the partition and checks the consistency of the resulting CSP : only sub-domains leading to consistent CSP are kept in the minimal domain. If a minimal domain is found to be empty, then the CSP is inconsistent.

4.2 Mixing timePoints and Variables

Among the extensions of the representation presented above, we have insisted on the necessity to express actions whose effects depend on their duration. It implies to be able to handle "transversal" constraints binding both timepoints and atemporal variables, i.e. coupling constraints between the atemporal CSP and the Time-Map.

To illustrate such a constraint let us consider a simple example dealing with the control of a robot : an action "Move" brings a robot from one point to another along an axis at constant speed. Thus the duration of the shift is proportional to the distance between the initial and final position of the robot : $Pos_f - Pos_i = speed * (t_f - t_i)$.

The first observation is that this kind of constraint can be splitted, each part being inserted in a specific Constraints Manager : the difference T between t_f and t_i can be introduced in the atemporal variables manager trough the constraints ($Pos_f - Pos_i = speed * T$) and the other way round, the difference D between Pos_f and Pos_i yields the constraints ($t_f - t_i = D/speed$) to be added to the Time-Map. Since both the Time-Map and the atemporal constraints network are dynamically constructed, we shall iterate such constraints posting during the search process. It should be noted however that the resolution of constraints problems computes minimal valuation domain. Thus from one step to another, the domain associated to a variable will shrink : consequently if two distance

Table 3. The Complete Propagation Algorithm

```
Complete_Propagation(){
   \*** computes minimal domains and ***\
   \*** returns true if the CSP is consistent ***\
   For each variable var
      AcceptedValues ← ∅
      Elts ← Partition(var)
      For each e in Elts
         If (unify(var, e) ∧ Global_consistency_checking())
            AcceptedValues.append(e)
         endIf
      endFor
      If AcceptedValues = ∅
         RETURN False
      Else
         Domain(var) ← AcceptedValues
      endIf
   endFor
   RETURN True
}
```

```
Global_Consistency_Checking(){
   \*** returns true if the CSP is consistent ***\
   var ← Choose One Non Instanciated variable
   Values ← Partition(var)
   While Values ≠ ∅
      elt ← pop(Values)
      If (unify(var, elt) ∧ Global_consistency_checking())
         \*** unify triggers local propagation ***\
         RETURN True
      endIf
   endWhile
   RETURN False
}
```

```
Partition(v){
   \*** This function returns the set of alternative ***\
   \*** disjoint domains for v ***\
   If there is at least one ≠ constraint on v then
      RETURN {{e}, e ∈ Domain(v)}
   Else
      Part ← {Domain(v)}
      LDep ← {D, ∃(v', D'), (v ∈ D ⇒ v' ∈ D') ∨ (v' ∈ D' ⇒ v ∈ D)}
      While LDep ≠ ∅
         KDom ← pop(LDep)
         NewPart ← ∅
         While (Part ≠ ∅)
            Elt ← pop(Part)
            NewPart ← NewPart ∪ (Elt ∩ KDom)
            NewPart ← Newpart ∪ (Elt ∩ ∁_KDom)
         endWhile
         Part ← NewPart
      endWhile
      RETURN Part
   endIf
}
```

constraints on the same variable are successively posted, the second one will discard its predecessors.

The second point we must focus on is the difference in handling disjunctions by the two constraints managers. As mentioned in the previous paragraph, domains associated to numeric atemporal variables are disjunctions of intervals. On the contrary the use of disjunctions in the time-map manager is forbidden. To bridge the gap between these two managers we will discuss more in-depth the time-map manager.

Basically this manager handles Simple Temporal Networks [Dechter 89]. It has limited expressivity but complete propagation can be computed in polynomial time thanks to the Floyd-Warshall algorithm, for instance. On the contrary, if we extend the expressivity to disjunctions of intervals (that is to say Temporal CSP or T-CSP), we are then confronted to NP-Hard problems [van Beek 89]. The solution retained in IXTET is to limit the expressivity so as to ensure completeness at every step of the planning process and to deal with disjunctions at the control level with explicit estimates and heuristics.

A possible solution to the coupling between the two CSPs would be to keep simple intervals inside of the temporal network, and to relax constraints inherited from the atemporal variables manager by approximating a disjunction of intervals to its lower and upper bounds. The propagation would then be incomplete because of the relaxation : an inconsistency may be detected only once the time-map manager is committed to instantiate a variable to a value that was not part of the initial disjunction on the atemporal manager's side. The handling of such a situation will be achieved at the search level that should backtrack in order to avoid the inconsistent constraint.

Another solution is to accept to pay the cost of handling disjunctions of intervals in the temporal constraint network. This would enable the search process to rely on complete propagation between the two constraints managers and to backtrack immediately if a constraint is introduced that gives rise to inconsistency.

We have chosen this second solution. The use of disjunctive constraints has long been avoided because it was known to be intractable in worse case. But recent experiment results prove that in most practical cases actual complexity of propagation of sets of intervals through arithmetic constraints such as the one we are concerned with is linear rather than exponential [Shapiro 99]. Furthermore this simplifies the control process, gets rid of heuristic evaluation and avoid backtracking, which our experiments proved to be a real bottleneck for the planner.

5 Discussion and conclusion

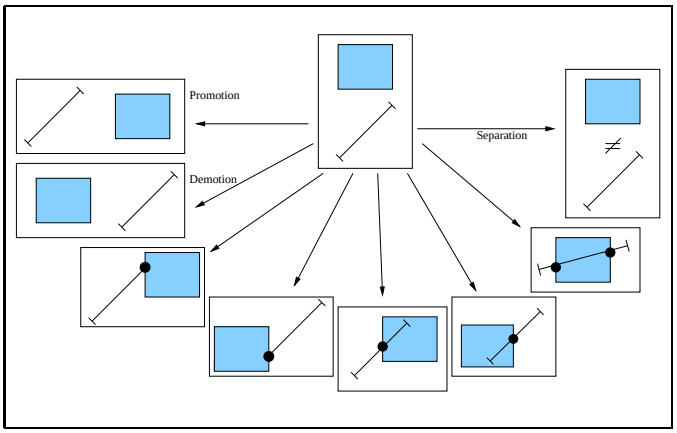
Most of the temporal planners that address the same goals as IXTET rely on the same kind of representation of the world. The most spectacular application in the field was the Remote Agent Experiment lead by the NASA : among the different technologies that were embedded in the remote agent architecture, the planning system PS had a crucial role. In RAX-PS, the world is also described through a set of temporally qualified state variables, some of which represent activities executed by a device : there is a strong parallel between a token associated to a literal and the invocation of an activity to be executed by the execution supervisor. A time point represents the moment at which the activity corresponding

to the new token starts executing. The representation in RAX-PS differs from the one in IxTeT on the following aspects. First in RAX-PS there is no explicit representation of events. The evolution of a variable (time line) is represented by a succession of temporal assertion (*hold*). This leads to slight differences in the search process since it is then guided by completely filling time-lines with assertions, the consistency between two successive assertions being defined by "compatibility rules". This brings out the second difference : in RAX-PS, there are no planning operators such as the *Tasks* mentioned above. The system is only concerned with attribute's value (which represent real procedures for the executive system) and benefits from rules to establish links between assertions along different time lines. Beyond those differences, RAX-PS relies, as IxTeT does, on a set of piecewise constant functions and on a CSP-based approach. It can benefit from our proposed extension. [1]

Among temporal planners, parcPLAN presents similar expressivity, and seems to offer quite interesting performance [Liatsos 99]. It relies on relatively similar representation and search strategy (constraint-based and least commitment). The main difference with IxTeT is the strong separation in the search process between goal achievement and scheduling reasoning. Here again piecewise linear functions could be beneficial.

Last but not least, [Penberthy 94] presented ZENO, a temporal planning system that offered rich expressivity to describe the dynamic of the world. Metric preconditions and effects as well as linear evolution were supported. However these features were supported without the benefit of constraints network methods that enable the search control to concentrate on establishing constraints instead of selecting exact values for variables. It should be noted that, as far as we know, Zeno has only been tested on simple toy domains.

Table 4. Threats concerning the extended hold predicate



[1] Following an interesting remark from an anonymous referee, we checked out in the available literature to what extent the temporal features described in this paper are supported in RAX-PS and we found no clear indication that this system handles temporal primitive other than piece-wise constant values.

An implementation of the extension presented in this paper is currently under process within IxTeT. We plan to compare it to the technique that approximates a *change* through multiple steps in complex domains. Another extension we are considering is a relaxed *hold* predicate : $hold(p : (v, v'), (t, t'))$ specifies that p remains bounded within interval [v,v'] during [t,t'[. Such a predicate is needed to provide better control over continuous evolutions : it enables one to express conditions that might be verified during an evolution, thus increasing possibilities to interleaved actions. The search control could be developed in the same way as for *change* predicate in order to cope with these new assertions. Once again the difficulty is pushed on the constraints solver. Table 4 sums up the possible situations to be examined to deal with such predicates.

References

[Chapman 87] D. Chapman. *Planning for Conjunctive Goals.* Artificial Intelligence, vol. 32, 1987.

[Dechter 89] Rina Dechter, Itay Meiri & Judea Pearl. *Temporal Constraint Networks.* In KR'89: Principles of Knowledge Representation and Reasoning, pages 83–93. Morgan Kaufmann, 1989.

[El-Kholy 96] A. El-Kholy & B. Richards. *Temporal and Resource Reasoning in Planning: the parcPlan approach.* In Proc. ECAI-96., 1996.

[Ghallab 94] M. Ghallab & H. Laruelle. *Representation and Control in IxTeT, a Temporal Planner.* In Proceedings of the International Conference on AI Planning Systems, pages 61–67, 1994.

[Kondrak 97] G. Kondrak & P. Van Beek. *A theoretical evaluation of selected backtracking algorithms.* Journal of AI, vol. 89, pages 365–387, 1997.

[Laborie 95] P. Laborie & M. Ghallab. *IxTeT: an Integrated Approach for Plan Generation and Scheduling.* In Proceedings ETFA-95, 1995.

[Liatsos 97] V. Liatsos & B. Richards. *Least commitment—an optimal planning strategy.* In Proceedings of the 16th Workshop of the UK Planning and Scheduling Special Interest Group, 1997.

[Liatsos 99] Vassilis Liatsos & Barry Richards. *Scaleability in Planning.* In Proc. ECP-99, pages 49–61, 1999.

[Muscettola 97] N. Muscettola, B. Smith, S. Chien, C. Fry, G. Rabideau, K. Rajan & D. Yan. *Onboard Planning for Autonomous Spacecraft.* In Proceedings of the 4th International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS 97), 1997.

[Penberthy 94] J. Scott Penberthy & Daniel S. Weld. *Temporal Planning with Continuous Change.* In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), vol. 2, pages 1010–1015. AAAI Press/MIT Press, 1994.

[Shapiro 99] Rony Shapiro, Yishai A. Feldman & Rina Dechter. *On the Complexity of Interval-Based Constraint Networks.* In MISC'99 Workshop on Applications of Interval Analysis to Systems and Control, 1999.

[van Beek 89] Peter van Beek. *Approximation Algorithms for Temporal Reasoning.* In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 1291–1296, 1989.

[Weld 94] Daniel S. Weld. *An Introduction to Least Commitment Planning.* AI Magazine, vol. 15, no. 4, pages 27–61, 1994.