Plans simulation using temporal logics

Eric Rutten, Lionel Marcé

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(1) 39 63 55 11

Rapports de Recherche

N° 1095

Programme 6
Robotique, Image et Vision

PLANS SIMULATION USING TEMPORAL LOGICS

Eric RUTTEN
Lionel MARCE

Septembre 1989

RR_1095

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Plans Simulation Using Temporal Logics

Eric RUTTEN, Lionel MARCÉ
IRISA / INRIA - Rennes
Campus de Beaulieu
F-35042 RENNES Cedex FRANCE
e-mail: rutten@irisa.fr, marce@irisa.fr

Publication Interne n° 487 - Juillet 1989 - 40 Pages

Abstract

In the field of telerobotics, an important part of the control of teleoperated systems and robots is in the hands of a human operator, who interacts through a global human-machine interface. An important component of this interface is a high-level representation of the system, enabling the operator to express what he wants it to do, and on which various kinds of treatments and reasoning can be carried out. One of these is the simulation at task-level of the plans of actions conceived by the operator, in order for him to evaluate the correspondence of their effects to his expectations.

We propose constructs for a language, allowing the writing of plans, provided with a clear, explicit control structure, disposing actions in time. The control operators are defined, as well as the actions and the environment on which they work, in terms of an unified formalism, where the temporal dimension, most important in a world featuring simultaneity and parallelism, is taken explicitely into account: an interval-based temporal logic. A simulation method, based on this model of the execution of plans of actions, takes advantage of the inferential and expressive power of the underlying logical formalism, and is illustrated by an example in space telerobotics. [1]

[1] This report is the text of a communication at the IJCAI Workshop on Integrated Human-Machine Intelligence in Aerospace Systems, held in Detroit, U.S.A., August 21 1989, augmented with an example, in the appendix.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

SIMULATION DE PLANS PAR LA LOGIQUE TEMPORELLE

Résumé

Dans le domaine de la téléopération, une part importante do contrôle des systèmes et robots téléopérés est dans les mains d'un opérateur humain, qui intéragit au travers d'une interface homme-machine globale. Un composant important de cette interface est une représentation de haut niveau d'abstraction du système, permettant à l'opérateur d'exprimer ce qu'il veut le voir effectuer, et surlequel diverses sortes de traitements et raisonnements peuvent être menés. Un de ceux-ci est la simulation, au niveau tâche, des plans d'actions conçus par l'opérateur, pour lui permettre d'évaluer la correspondance de leurs effets à ce qu'il en attend.

Nous proposons un langage permettant l'écriture de plans munis d'une structure de contrôle claire et explicite, diposant les actions dans le temps. Les opérateurs sont définis, ainsi que les actions et l'environnement sur lequel elles travaillent, en termes d'un formalisme unifié, où la dimension temporelle, des plus importantes dans un monde montrant de la simultanéité et du parallélisme, est prise en compte explicitement: une logique temporelle d'intervalles. Une méthode de simulation, fondée sur ce modèle de l'exécution de plans d'actions, tire avantage de la puissance d'inférence et d'expression du formalisme logique sous-jacent, et est illustrée par in exemple en téléopération spatiale. [2]

[2]Ce rapport reprend une communication au *Workshop on Integrated Human-Machine Intelligence in Aerospace Systems*, tenu à Detroit, U.S.A., le 21 Août 1989, dans le cadre de l' *IJCAI'89*, auquel un exemple a été ajouté en annexe.

Contents

1 Introduction

We propose a model for the representation of the execution of plans. They are composed of actions, linked together by control operators, and are performed on an environment, that changes as an effect of their execution.

The motivation of this work, and its general frame, is **operator assistance in telerobotics** (section 2.) Telerobotics are characterized by the presence of a human operator, who commands the system in control modes ranging from manual to automatic. The interfacing between the human and the machine must feature a language in which to express what is to be executed. Decision assistance treatments, such as the simulation of their effects, are applied on plans specified in this language. This entails the need of a representation of the system and its environment, based on an underlying formalism, on which these treatments can be made.

For that, we have chosen a particular type of logic: Allen's interval-based **temporal logic** (section 3.) It brings us the basic features of our representation, including an explicit treatment of the temporal dimension of the modeled world. This is particularly useful, as we are interested in a changing environment, where events can take place parallelly, have a duration, and other temporal characteristics.

On the basis of this formalism, we define **plans and actions** in terms of the temporal logic (section 4.) Actions are seen as the basic changes in the world situation, and plans consist in linking actions together in a control structure. The structure is determined by control operators, that specify how actions are placed in time. This language enables the writing of complex, yet clearly structured plans.

The **simulation** of such plans consists in determining the effects of the actions they are composed of, and the way they take place in time (section 5.) It gives information from which the operator can examine the previsible effects of his plan, and their correspondence to his expectations. The simulation process is illustrated by the example of a space station.

An overview of **related work** shows that other formalisms of plans and actions, and other approaches to operator assistance exist (section 6.) The **perspectives** of this work can be found in several directions (section 7), and concern the model of time and that of action, the extension of the control structures, as well as the application of the plan execution model to other situations, or the use of the simulation method.

2 Operator Assistance in Telerobotics

Telerobotics Telerobotics are an example of system for which there is a human intervention, at each stage of the *perception-decision-action* control loop. The motivation of the presence of a human operator relies on his natural intelligence and skills, which provide more flexibility in the system operation, than what can be achieved through complete automation, in its present state of the art. The telerobotics approach constitutes not only an alternative to the autonomous agent one, but an intermediary stage on the way to it, where concepts can be experimented and refined. This means that the control is shared between the human operator and the machine in a variable proportion: a variety of control modes must be available to him, ranging from manual to automatic, and enabling him to make his interventions at any level, but only when it's needed.

Human-machine interface This variety of control modes entails the need for a human-machine interface, assisting the human operator in his task of supervision and control of the operated system. This assistance is usefull at all stages of the *perception-decision-action* loop. *Perception* of the state of the environment that is to be worked upon, can be asssisted by a convenient treatment of the data acquired from it (e.g. through data fusion, synthetic visual feedback). The plans of *actions* to be performed, are controlled through an execution monitoring system, that has to manage the interactions between operator and machine at execution time, i.e. his interventions as well as the information coming from the system.

Decision assistance can take several forms, one of them concerning the preparation of the action stage. Indeed, the constraints imposed by the environment can impose a careful behavior, e.g. in hostile environments like nuclear plants or space stations. In these cases, the irreversibility of the execution of certain actions, and difficulty of recovering from a failed execution, imply the need of thinking twice before acting, for the operator. This second thinking of the operator is where an assistance is needed, that can consist in a simulation tool, allowing to study previsible executions of a plan before starting executing it.

Plan language The communication with the system requires a language, with which the operator can specify, read and follow the control of plans of actions. The plans control structure, indicating the flow of execution of the actions, is what gives the language its expressive power. Particular control operators enable the specification of particular temporal dispositions of actions. Their readability, which is linked to the clarity of their definition, is particularly important to the human operators, who need that complex plans remain clear, so that they don't loose the thread of their execution.

Most work on plans and actions representation has been made with plan generation as its goal, and the question of knowing what the control structures of plans should be has not found an unanimous answer by now [12]. Although a plan for a robot is not just like programming a computer, we place ourselves in an approach where a human operator has to decide of and conceive the plan, and then to supervise its execution. The plans we are talking about are written by hand, this is why we use the concept of control structure here, as an explicit support to conception and supervision. We therefore propose a set of control operators. Some simple ones express, in an imperative way, the sequencing, parallelization and conditionality of actions subplans. Others represent operators that are particularly needed for the representation of realistic robots, and are more specific to the domain.

Simulation We are here particularly interested in the simulation of plans, that gives as result a representation of what effects are entailed by the execution of the plan. This must be made in a form readable and exploitable by the human operator, so that he gets help from it when deciding on the construction of the plan. The way this simulation takes place, is by transcribing the effects of the actions on the representation of the environment, in which changes are happening along time. Their execution follows the control structure of the plan they compose. This supposes that the representations of the plan structures, of the actions and their effects, and of the environment are compatible.

This is achieved by using the same, unified underlying formalism as a support for representing as well the characteristics of the environning world, as the actions and the changes they entail, and the control structures and the way they determine the temporal disposition of actions. This formalism is mainly inspired by Allen's interval-based temporal logic [1].

Temporal logics The advantage of using a logical formalism for such a knowledge representation issue, is that it provides us with a formal language, with which unequivocal specification can be made. It also provides us with an explicitly expressed reasoning capacity, knowledge and its treatment being expressed in the same natural way. The use of a temporal logic is justified by the parallelism of the plans that we want to describe, and the possibility of events happening in the world external to the robot's control. Time is a concept deeply linked to all that has to do with change and simultaneity. Making this temporal information explicit, and providing us with tools to manipulate it, is how temporal logics can achieve representation of time-related knowledge.

Allen's formalism presents a model of time that is natural, and relatively close to human reasoning, as to the expression of relative placements of events, temporal relations between them, truth of facts over time, ... It also offers the possibility of introducing quantitative information, in the form of dates and durations, which must be taken into account as well as qualitative or symbolic information. It has been used, amongst others, for plan generation purposes [2], and has motivated many studies on its various aspects: theoretical foundations, implementation and complexity issues, applications, ... [6, 13, 11].

3 Temporal Logics

The attempt to capture the notion of time into a logical frame has given way to the building of many different formalisms, among which modal tense logics, instant-based formalisms, interval-based ones, ... [14]. We are mostly interested in interval logics of the type developped by Allen [1], because of their expressivity concerning duration and parallelism. The way they can be linked with the notion of instants also interests us [3]. A generalization of some of their features proposed by Shoham [11] is useful to us.

Intervals We will adopt the notion of intervals being "chunks" of time, between which relations have been determined by Allen, and are shown in figure 1. These thirteen relations capture all the possible relative positions for a pair of intervals. Moreover, a transitivity table has been built on these bases, allowing to determine, knowing two relations r_1 and r_2 between three

intervals I_1, I_2 and I_3, i.e. $I_1\ r_1\ I_2$ and $I_2\ r_2\ I_3$, a third relation r_3 obtained transitively: $I_1\ r_3\ I_3$. These relations can be grouped in a disjunctive relation if the precise relation is not known; for example: $((I_1\ r_1\ I_2) \vee (I_1\ r_2\ I_2))$ can be written: $(\ I_1\ (r_1\ r_2)\ I_2\ )$.

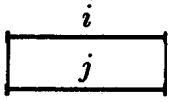
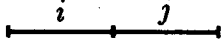
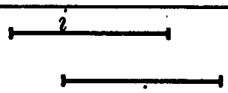
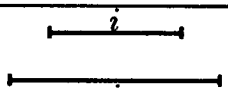
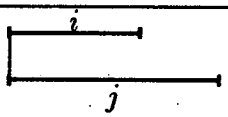
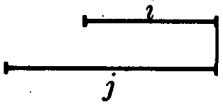
relation	inverse	graphical exemple
i before (<) *j*	*j* > *i*	
i equals (=) *j*	*j* = *i*	
i meets (m) *j*	*j* mi *i*	
i overlaps (o) *j*	*j* oi *i*	
i during (d) *j*	*j* di *i*	
i starts (s) *j*	*j* si *i*	
i finishes (f) *j*	*j* fi *i*	

Figure 1: Allen's 13 possible relations.

Temporal facts As summarized by Shoham, facts represented in classical first order logic by predicates, are here "reified" into temporal facts. Those take the form: *true(I, P)*, meaning that the fact P holds on the interval I. The rules of classical logic are assumed to hold, in the way established by Shoham [11], stating that, for example, $true(I,P_1 \wedge P_2)$ is satisfied if and only if $true(I,P_1)$ and $true(I,P_2)$ are. The treatment of negation gives way to an alternative discussed by Shoham, and for this article, we will choose to say

that *true(I,¬P)* holds if and only if, for no subinterval I' of I, we can have *true(I',P)*.

Classification by inheritance Shoham suggested another way of getting new temporal information from known facts, by classifying temporal facts according to the relation of their truth over one interval to their truth over other intervals [11]. Exemples of classes are: *downward-hereditary* (e.g. "The robot travelled less than two miles.", when true over an interval, is true over all its sub-intervals), *upward-hereditary* (e.g. "The robot travelled at the speed of two miles per hour.", when holding for all the proper sub-intervals of a nonpoint interval, holds for the nonpoint interval itself), *liquid* : both *upward-hereditary* and *downward-hereditary* (e.g. "The robot's arm was in the GRASPING state"), *solid* (e.g. "The robot executed the NAVIGATE procedure (from start to finish)" never holds over two properly overlapping intervals), ... Other classes can be imagined and defined through the particular relations between intervals of truth. The temporal facts (*true(I,Fact)*) described in the previous paragraph are *liquid*. The *solid* class can be used to describe the execution of an action or a plan (e.g. by *exec(I,plan)*.)

We thus are provided with a representation formalism allowing us to describe a world, especially as temporal information is concerned, and with manipulating rules with which further information can be obtained from the one already explicited, i.e. reasoning can be made. We will now see how we intend to use this formalism in order to describe the temporal involvement, as well as the logical behaviour, of plans provided with a control structure.

4 Plans and Actions

As in our approach, a human operator is involved in the decision process, i.e. he writes himself the plans for the robot, we have to provide him with a language in which to express the plans. This language must allow him to build complex, yet clear, control structures. Its basic elements are the *primitive actions*, which are defined by the changes entailed by the execution. These actions can be grouped into structures determined by *control operators*, specifying in what way the actions they frame will take place in time.

Primitive Actions A classical representation for an action is that adopted for *STRIPS* and its successors, where an action is represented through three sets: the facts that are needed to hold, for it to be executed (the *precondition*-list), and the effects it entails in its environment, positive (the *add*-list, of facts made true as a consequence of the action) as well as negative (the *delete*-list, of facts made false.) We however want to add a temporal information to this: we therefore associate, with each action occurence or execution, an interval, the extent of which is given by the duration of the action. An action is noted as shown in fig. 2, where an example is given, of the action of an arm A, *taking* a piece P_1, *fixed to* a piece P_2. A representation of the example action of fig. 2 in time is given by fig. 3.

representation	example
action(name(parameters), duration, preconditions, negative effects, positive effects).	*action(take(A, P_1, P_2), 2 mn, [fixedto(P_1, P_2), available(A),accessible(P_1)], [fixedto(P_1,P_2), available(A),accessible(P_1)], [accessible(P_2),held(P_1, A)]).*

Figure 2: Actions representation.

As Vere did [15], we make the "changes on termination" assumption, deciding that all the changes entailed by the action occur at the end of its occurence. Nevertheless, the preconditions have to be holding on all the interval of the action. The "changes on termination" assumption, however, is not limitative: the definition of *compound actions*, presented further, allows to define actions having effects in an other way.

It takes place in time in a way described by the following formula, where *lasts(i,d)* makes the correspondence between an interval i and its extent d:

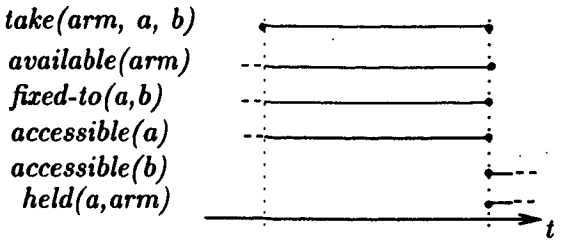


Figure 3: Example of an action: *take(arm,a,b)* in time.

$$
\begin{aligned}
exec(I,act) \Longleftrightarrow\ & action(act,\ d,\ P,\ C,\ A) \\
& \wedge\ lasts(I,d) \\
& \wedge(\forall p \in P)\ true(I,p) \\
& \wedge(\forall c \in C)(\ (\exists I_c)(true(I_c,c) \Longrightarrow (I_c\ (\text{b m o s d f fi} =)\ I)) \\
& \wedge(\forall a \in A)(\exists I_a)(true(I_a,a) \wedge I\ (\text{d s oi m})\ I_a).
\end{aligned}
$$

At the moment of simulating such an action, the deduction tools of the temporal logics will be useful to find out wether the preconditions hold over the interval, and how the effects interact with other facts in the world representation.

Plans A plan, as said earlier, is considered here as a set of actions, provided with a *control structure*. A plan is then composed of subplans, which are plans themselves, recursively, down to the *primitive actions*. The basic constructs of the language are:

sequence : noted *seq(subplans list)*. For a subplans list $[P_1|P]$, where P_1 is the first subplan of the list , and P the remainder of the sequence, this control operator states that the subplans of the list are executed one after the other, in the order if the sequence(fig 4.)

$$
\begin{aligned}
& exec(I,\ seq([P_1|P])\ ) \Longleftrightarrow \\
& \quad (\exists I_1)(\exists I_P)(I_1\ \text{s}\ I \wedge exec(I_1,P_1)\ \wedge I_1\ \text{m}\ I_P \\
& \quad \wedge\ exec(I_P,\ seq(P)\ ) \wedge I_P\ \text{f}\ I).
\end{aligned}
$$

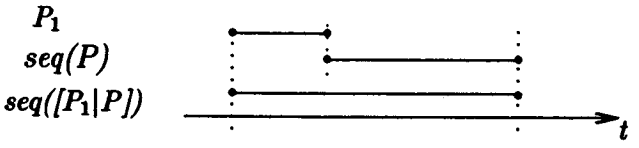


Figure 4: A sequence in time.

parallelism : noted *par(subplans list)*, where the list of subplans contains the plans constituting each a branch concurrent to the others. In this construct, all branches B start together, and the parallelism ends when all branches have ended, i.e. with the longest lasting branch, B_P (fig. 5.)

$$exec(I,\ par(Branches)\ ) \Longleftrightarrow$$
$$(\exists B_p \in Branches)\ (\ exec\ (I, B_p)$$
$$\wedge(\forall B \in Branches - \{B_p\})(\exists I_B)\ (\ exec(I_B,\ B\ ) \wedge I_B\ (\mathrm{s} =)\ I\ )\ ).$$

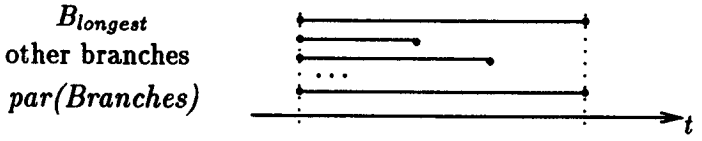


Figure 5: Parallelism in time.

conditionality : noted $cond(C,P_{true}, P_{false})$. The condition C is evaluated on the beginning of the interval, and following the result, the corresponding subplan is executed (fig. 6.)

$$exec(I,\ cond(C,P_{true}, P_{false})) \Longleftrightarrow$$
$$(\exists I_c\ )\ (\ (true(I_c, C) \wedge (I_c\ \mathrm{s}\ I \wedge exec(I,\ P_{true}))$$
$$\vee\ (\ true(I_c, \neg C) \wedge (I_c\ \mathrm{s}\ I \wedge exec(I, P_{false})\ )\ ).$$

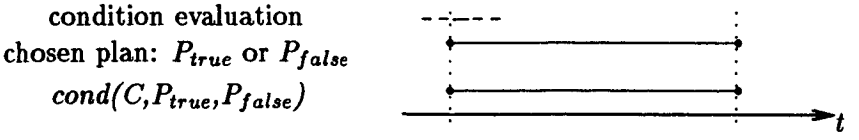


Figure 6: Conditionality in time.

reactive operators The operators previously seen are adapted from the classical computer programming style. In the robotics field, as well as in real-time oriented problems, another kind of operators is necessary, to express other temporal schemes. Generally, we need to express that something can happen "as soon as" something else does. This can be characterized in several ways. We have considered some of them, and the similitudes between them.

reactivity : *as-soon-as(Condition,Plan)*, where the starting of a plan is linked to the satisfaction of a condition. The way it takes place in time is illustrated in fig.7. This is the most basic form of reactivity, and can be seen as the basis on which others can be constructed.

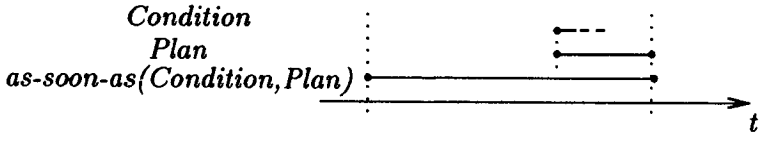
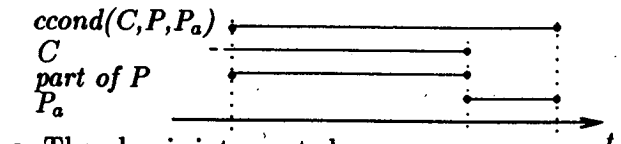
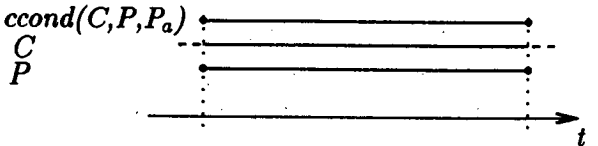


Figure 7: Reactivity in time

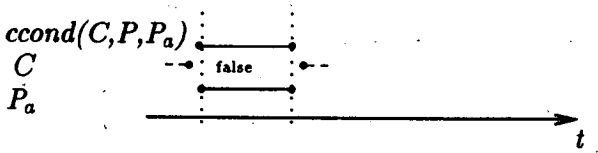
continuous condition : *c-cond(Condition, Plan,* $P_{alternative}$*)*, where the actions in the *Plan* are executed as long as the *Condition* holds. The *Plan* can be interrupted at the failure of the *Condition*, and the remainder of it, i.e. the part not executed at that time, is withdrawn. A possibility is given, of specifying an alternative plan: $P_{alternative}$, that is executed as a kind of recovery procedure.



a. The plan is interrupted.



b. The plan is executed completely.



c. The alternative plan is executed.

Figure 8: Continuous conditionality in time

This case is shown in fig.8*a*. If the *Condition* holds on an interval containing that of the *Plan*, we are in the case of fig.8*b*. When the *Condition* does not hold, only the alternative plan is executed (fig.8*c*.)

elastic actions : *as-long-as(Plan,Elastic-Action)*. In this construct, we specify that the duration of the *Elastic-Action* is determined by that of the *Plan* to which it is associated. This entails the need of having a special type of actions, different from those seen before, in that no duration would be associated to them *a priori*.

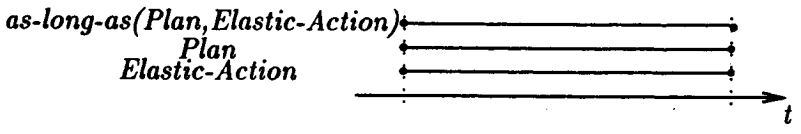


Figure 9: An elastic action in time

Examples of actions of that kind are: *hold* something, or *move* in some direction.

similitudes All these operators are based on the same idea, that their temporal definition is not always self-contained, i.e. that the disposition in time of the actions sometimes needs to be determined by external circumstances. They obviously share some characteristics in common, while expressing each a different structure.

The continuous conditional can be used to express something close to the basic reactivity, by writing: *ccond(¬Cond , wait, Plan)*, where *wait* can be seen as: *while(true, delay)*, *delay* being an action having a duration, but no preconditions, nor effects. Elastic actions could be interpreted as a kind of parallelism, where the branches would all have the same duration. It also corresponds to a continous conditional on the execution of the plan: when it is not true any more that the plan is executed, i.e. at its end, the elastic action is interrupted, the alternative plan being empty. An elastic action having a precondition can be seen as a continuous condition.

These apparent similitudes need to be characterized, and could lead to a classification on reactive behaviours (section 7.)

Compound Operators and Actions We have a set of basic constructs, which are defined to be assembled in order to form plans. In this task, it is helpful to be able to define new items from the basic ones, and to re-use them. This can be done here in two ways.

compound operators From these operators, others, more customized, can be constructed, like, for example, *conditional iteration*, defined recursively as:

while(*condition, iteration-body*)
≡
cond(*condition*,
seq([*iteration-body*,
while(*condition, iteration-body*)]),
nothing).

where *nothing* is a null action, taking no time, requiring no precondition, and entailing no effect.

compound actions In the same manner, more complex actions can be defined, using a plan, in the following way:

compound-action(name(arguments) , plan).

This allows to consider a sort of macro-actions, with effects dispatched along their duration, or depending on the context where they are executed. For example, a way of defining an action a realizing effects on an interval as in fig.10, is to decompose it in the following way: *compound-action(a , seq([a_1, a_2]))*, where we have an action a_1 such as: *action(a_1, d_1, [p_1,p_2], [p_2], [p_3])*, and a_2: *action(a_2, d_2, [p_1], [], [p_2])*.

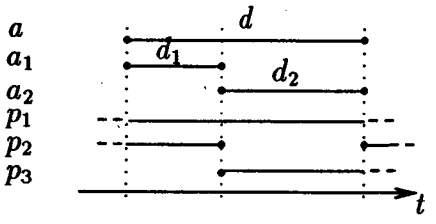


Figure 10: A compound action in time: *compound-action(a , seq([a_1, a_2]))*.

The total duration of action a is then $d = d_1 + d_2$, as a_1 and a_2 are in sequence: an action having its effects on beginning would be of this form, with $d_1 = 0$.

Example An example can be given in a space station environment, as shown in fig. 11. The world is composed of two manipulation arms, *arm1* and *arm2*, both *available* in the initial situation. There is a structure of elements of a extension of the *station body*: they can be *fixed to* each other, the whole being fixed to the *station body*. Only the *end element* of the structure is *accessible* to an arm. There is also a *new element*, that is also accessible, fixed on a *rack*.

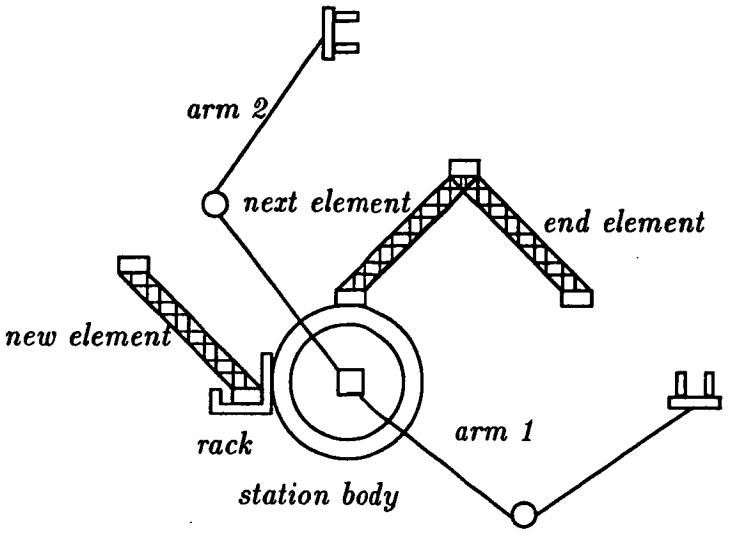


Figure 11: An example in a space station

The arms can execute two kinds of actions on the elements: to *take* an element from another one, defined as seen before in fig. 2, or to *place* it on another one, defined by:

```
action(  place(Arm, Piece1, Piece2),
         1.5 mn,
         [accessible(Piece2), held(Piece1, Arm)] ,
         [accessible(Piece2), held(Piece1, Arm)] ,
         [available(Arm), fixedto(Piece1, Piece2),
               accessible(Piece1)] ).
```

In order to insert a *new element* between the *end element* and the *next element* in the chain, a possible plan that an operator can write is:

```
seq([  par([  take(arm1,endelement,nextelement),
              take(arm2,newelement,rack)
              ]),
       place(arm2,newelement,nextelement),
       place(arm1,endelement,newelement)
       ]).
```

The simulation of this plan, on the initial situation described above, gives as result the trace shown in fig. 12. For further use in other plans in that world featuring element insertions, this plan can be defined as a compound action:

```
compound-action( insert(Newelement),
                 seq([ par([ take(arm1,endelement,nextelement),
                             take(arm2,Newelement,rack)
                             ]).
                       place(arm2,Newelement,nextelement),
                       place(arm1,endelement,Newelement)
                       ])
               ),
```

where the *Newelement* is the argument to this action.

The language presented enables the writing of structured plans, composed of actions defined by their effects on the world they are executed in. The temporal dimension of these actions and plans having been specified, we will now see how simulation on these bases is achieved.

5 Simulation

Our approach is concerned with simulation. The difference to plan generation is that the plan is a given data to the simulator, and the results that are expected, are the state reached as a consequence of the execution of this plan. Its use for the human operator, is to allow him to evaluate the correspondence between the plans he wants to give to the robot, and the effects that he expects from their execution on the environment. The simulation of a plan then consists, given a representation of the world to which it is applied, in a modification of this representation by transcribing onto it the effects of the actions of the plan, taken following the control structure leading their flow.

The simulation process The world representation consists in a set of temporal facts of the kind described earlier. A set of predefined actions is at the disposal of the planner, who can build a plan, constructed with control operators like those seen before. This plan is given as input to the simulation process, that begins with decomposing it down, to find the set of

the primitive actions that should be simulated first, because of their position in the control structure (e.g. those beginning a branch, a sequence), the rest of the plan being left in its original state. Then, primitive actions in the set are simulated one after the other, in a succession determined by the control structure of the plan. As the control flow advances in the structure, other primitive actions are added to the set, while the plan is "unrolled" progressively.

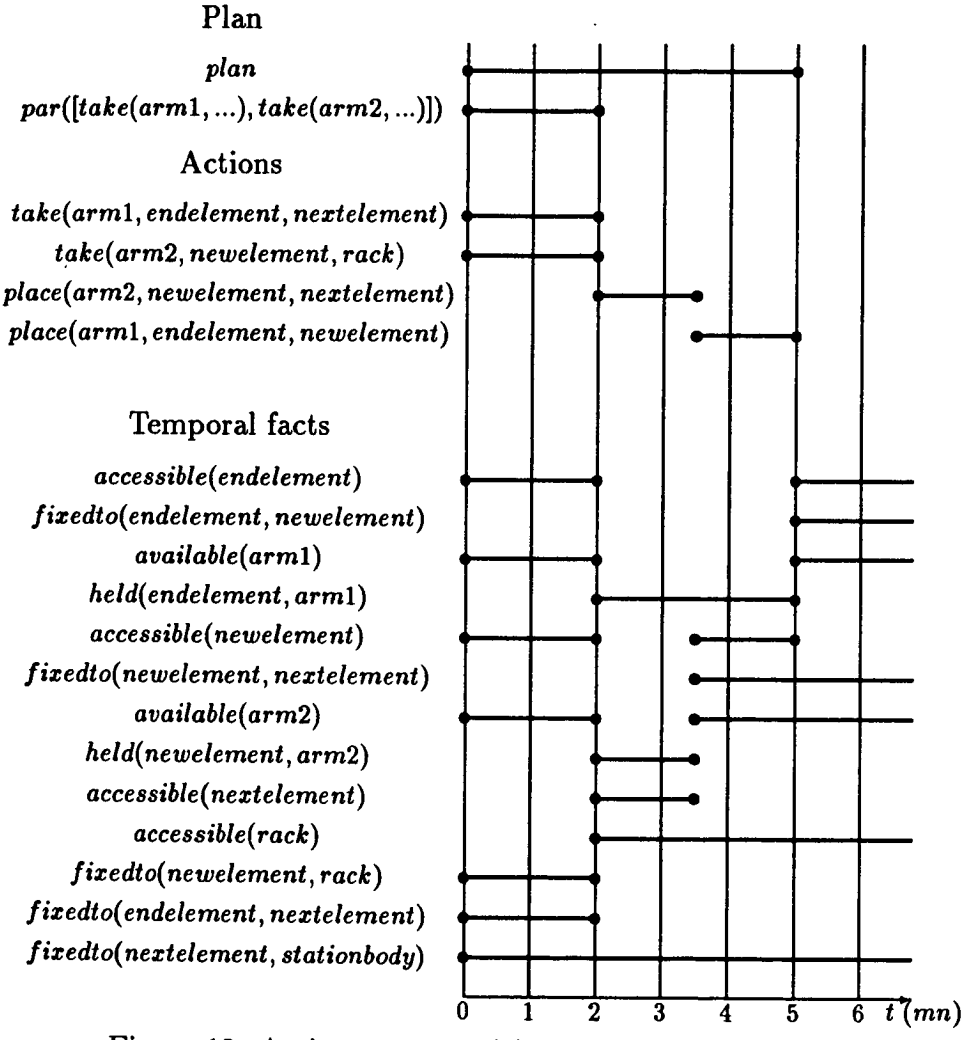


Figure 12: Actions, temporal facts, and associated intervals.

Primitive Actions The simulation of each primitive action, on an interval of which the extent is given by the duration, consists in: verifying the preconditions; transcribing the effects in the world representation.

At this stage, the temporal logic can be used to check the coherence of the new constraints introduced with the others, or to verify the compatibility of the positive effects with the temporal facts according to rules specified in the world representation. An *incoherence* is said to be encountered in the simulation, if either a precondition fails to be satisfied, or a contradictory constraint or an incompatibility of the effects is detected.

Plans The simulation of a plan then consists in the succession of simulations of the primitive actions they contain, following the order specified by the control structure. The task of simulating can be described as that of, in a loop terminating when the plan is through, choosing the next primitive action to be simulated, determining its associated interval, and treating its preconditions and effects. This happens in a way guided by the definitions of the control operators given in section 4. Compound operators and actions are rewritten, following their definitions in terms of basic elements, from their original form in an expanded form, down to basic plans, and simulated in that state.

An interesting case is that of orderings between actions that are not determined: their intervals have a relative position being a disjunctive relation. In this case, the simulator will try each relation in the disjunction, "forcing" the intervals into an order, and simulating further this possibility, until reaching the end of that simulation. Then a backtracking takes place, back to the last choice made, where, if another relation of the disjunction is left untried, it is taken as a new choice, and the simulation is resumed from this point.

Results In this way, the tree of possible executions of the plan, each one related to a possible disposition of the actions in time, is explored completely. The progression in a branch of this tree stops in two cases:

- either the end of the plan is reached: we then have the result that it is executable, and that the consequent state of the world is the one reached at this point;

- either an *incoherence* is encountered, i.e. a precondition to an action fails to be satisfied, or the effects of an action are incompatible with some fact: the result then consists in the verdict of unexecutability of the plan, with some information on the reasons of this failure, in order to help modifying the plan so that it would give a success.

To summarize, a plan built as seen before can be simulated with regard to its effects on the environment being worked upon, and also to the internal coherence of the dependencies between its actions, along its control structure. By exploring the possible consequences of that plan, it provides an operator with an assistance for the evaluation of the correspondence between the plan and what is expected of its execution.

6 Related Work

As we are working at the meeting of several domains, aspects of our work can be compared to other works in different fields. From the point of view of the temporal formalism and action representation, related works can be found in the field of knowledge representation using logics. These works cover the theoretical and general issues of temporal reasoning and planning. On the robotics side, works exist on the specification of control structures for task-level plans. They try to identify what kinds of behaviours are expected from a robot, and how to express commands to obtain them. Operator assistance studies, which are a basic motivation of our work, have given way to approaches corresponding to ours, while using different techniques.

Time and actions representations The representation of time that we have chosen, Allen's temporal logic, has been the object or starting point of many studies, by T. Dean, R. Pelavin, J. Koomen, E. Tsang, and others. Amongst them are Ghallab e.a. [6], who have developed algorithms for consistency maintenance in a time points lattice, in a system called IxTeT. They use it as a support for planning and execution control. Their model of actions is declarative, in the sense that preconditions and effects are specified as predicates associated with interval relations, situating them with regard to the action. A task is a set of such actions, and conditions on the environ-

ment, linked together with interval relations as well. The difference with our approach is that they don't use imperative control structures.

Another model for time and action is that of Sandewall e.a. [10]. It takes parallelism and lasting effects into account, in the frame of the Explicit Temporal Logic. Actions are seen as preconditions (true until the beginning of the action), postconditions (true starting from the end of the action), and prevail conditions (true on the duration of the action.) A plan is an action structure, consisting in a partial order between begin and end points of actions it contains.The prevail condition concept can be linked to our model by using the *compound-actions*. As for Ghallab e.a., but in a different way, the control structure of the plans is of a declarative kind.

Control structures and language In computer science, programming languages rely traditionally on *seq*, *par*, and *cond*,with also iteration in diverse forms (the star *, *while* and *repeat*, *do*, ...) An application in command of robots, using natural language, has lead Michalowski e.a. [8] to define a set of seven control structures: the sequence (*seq*) , parallelism (*par*) , condition (*if ... then ... else ...*), and iteration (*repeat ... until ...*), and three others, much more particular to robotics: *do ... (until ...)*, comparable to our elastic actions, *when(Condition, Plan)*, and *whenever(Condition, Plan)*, corresponding to reactivity, *when* meaning*as-soon-as*, and *whenever* being an endless iteration of *when*.

Operator assistance Guy Boy e.a. chose to give this assistance "on line", by a system following the operator step by step, and looking round at the changes of the environment at the same time [5]. Such a system provides advices or warnings, guiding the operator and avoiding his overlooking situations requiring his attention. In this case, the work provided consists in reasoning about a changing world, making assumptions on the base of incomplete information about it, and so forth. Formalisms on which to establish such reasoning are non-monotonic logics, that can be used in the frame of a blackboard architecture. Our approach consists in providing the operator with an assistance tool for the preparation of a mission or part of it, before its actual execution.

Artificial intelligence The most commonly allegated perspective, in the field of artificial intelligence in general, is that of providing an agent with total autonomy. An automatic reasoning capacity is essential to that purpose, and that is an essential motivation for the research in logical formalisms applied to artificial intelligence.

We chose to keep a place for the human operator's natural intelligence in the loop. A reason for that is not only that it allows to make things that are not possible automatically, by now. Another reason is also that, even if autonomous robots can operate alone, at the stage of their conception, an interface has to exist between it and its conceptors.

7 Conclusions and Perspectives

Starting from basic concepts, in different disciplines, that we linked together [9], we have proceeded to the present state, where several directions of extension interest us, concerning the different aspects of our approach.

Present state This article presents a represention of the execution of actions of robots, at an abstract level, along time. These actions are organized into plans, built with the help of control operators allowing the construction of complex, yet clear, structures. The world is represented, using a temporal logic enabling us to deal with situations involving duration and parallelism. The motivation of the application of logical formalisms to this kind of problems lies in their providing us with a clear knowledge representation frame, as well as tools for manipulating this knowledge, i.e. reasoning. Our approach consists in applying these formalisms to simulation, as a hopefully profitable alternative to the more classical, yet difficult, plan generation paradigm. This leads us to make the link between the temporal logics and a structured language featuring control operators. In the process of defining such operators, in a clear and unambiguous way, advantage can be taken from a formal model, as a support for the study of their characteristics. We confront our formalism to an experimental situation, that inspired the simple example of this article: an arm servicer on a space station. It is provided to us by MATRA-Espace, where studies are made, in the frame of a teleoperated arm system project [4], for the European Space Agency.

Language The model used is extendable with control operators more specific to the robotics world, and not derivable from the basic ones (synchronization on the satisfaction of a condition, ...), other kinds of actions (non-deterministic in their consequences, ...) Other operators, as we have seen in section 4, are more than useful to describe or specify plans of actions, dealing with a realistic view of robots. Those are mainly concerned with the reactive aspect of a robot programming. Several ways exist, to describe a reactive behaviour, but it not obvious wether they are not redundant in some way. Trying to classify them would be a good thing, in which the underlying temporal formalism could help. In order to take the presence of a human operator in telerobotics into account, an operator representing an *alternative*, that will give place to a choice by the operator at execution time, should be present also [7].

Knowledge representation The model of time is to be extended so as to take into account a certain imprecision on instants. We represent these instants with *(min, max)* couples, that we generalize to ordered lists of possible values. A further generalization would be to consider tolerance intervals, in which instants would be known to be, but with no further information than their boundaries. With the definition, on these tolerances, of an order relation and a sufficient algebra, we then can have multiple possible orderings, corresponding to the disjunctive temporal relations between intervals.

At each uncertainty as to the order between two instants, time can branch, following the possible time lines. To have access on these different branches, and to express oneself about the relations between them (e.g. the truth of a property on all future branches), modal temporal logics can be of help, and introduce a stronger expressivity. The branching between possible evolutions could also come from the actions: considering actions with non-deterministic effects, or of flexible durations (e.g. keeping doing something.)

Operator assistance The simulation method presented here should be applicable to domains other than robotics, where situations, involving time and parallelism, could be represented in the chosen formalism. The logical nature of the temporal formalism makes that the simulator produces more than a trace of the plan execution: it produces a factual knowledge base, which can be questioned in various ways. The interest of simulation can be

seen from different points of view:

- given a configuration of a system, and a reached state of its characteristics, it allows to test and refine a plan in order to make its effects correspond to the goal aimed at;
- given a configuration of a system, and a plan, the effect of which is known to achieve a certain goal, it allows to check for needed characterics in the state of the world, and to add preparation operations to the standard plan, if necessary, in order to make it applicable;
- given a state of the world, and a plan to be executed by a system, it can help checking what its configuration should be like, in order to execute the plan so that the state of the world would evolve in the intended way.

These three ways of considering simulation are very general, and a closer study of the use of simulation, its techniques, and the way the information it provides is used would be of interest.

Embedding the simulator in a more general and global operator assistance system links it to the other works in our team, which deal with formalisms for execution control, interactive geometric model acquisition, and sensor data fusion for visual feedback.

References

[1] J.F. Allen. An interval-based representation of temporal knowledge. In *Proceedings of the IJCAI '81*, pages 221–226, Vancouver, 1981.

[2] J.F. Allen, J. Koomen. Planning using a temporal world model. In *Proceedings of the IJCAI '83*, pages 741–747, Karlsruhe, August 1983.

[3] J.F. Allen P.J. Hayes. *Moments and points in an interval-based temporal logic.* Technical Report TR 180, University of Rochester, December 1987.

[4] G. André, T. Blais. Space teleoperation and control concept, experimental evaluation for the Hermes robot arm (HERA). In C.A. Mason, editor, *Proceedings of the International Symposium on Teleoperation and Control*, Bristol, England, 12-15 July 1988.

[5] G. Boy, N. Mathé. Using non-monotonic reasoning for operator assistant systems in reactive environments. In *Proceedings of the ESA-ESTEC Workshop on Artificial Intelligence Applications in Space Projects*, Noordwijkerhout, The Netherlands, November 15-17, 1988.

[6] M. Ghallab, R. Alami, R. Chatila. Dealing with time in planning and execution monitoring. *Robotics Research, R. Bolles (Ed.), MIT Press*, 4, 1988.

[7] P. Gravez. *Etude d'un système de supervision pour la téléopération assistée par ordinateur.* PhD thesis, Université de Lille, 1988.

[8] S. Michalowski, C. Crangle, L. Liang. A natural-language interface to a mobile robot. In *Proceedings of the Workshop on Space Telerobotics*, JPL publication 87-13 vol. II, NASA, Pasadena, California, July 1, 1987.

[9] E. Rutten, L. Marcé. Temporal logics meet telerobotics. In *Proceedings of the NASA Conference on Space Telerobotics*, NASA, Pasadena, California, Jan. 31- Feb.2, 1989.

[10] E. Sandewall, R. Rönnquist. A representation for action structures. In *Proceedings of the AAAI '86*, pages 89-97, Philadelphia, August 1986.

[11] Y. Shoham. *Reasoning about change : Time and causation from the standpoint of artificial intelligence.* PhD thesis, Yale University, December 1986.

[12] W. Swartout. DARPA Santa Cruz Workshop on Planning. *AI Magazine*, 9(2):115–131, Summer 1988.

[13] E. Tsang. Time structures for A.I. In *Proceedings of the IJCAI '87*, pages 456–461, Milano, August 1987.

[14] R. Turner. *Logics for artificial intelligence.* Ellis-Horwood Pub.Co., 1984.

[15] S.A. Vere. Planning in time : windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(3):246–267, May 1983.

A The Space Station Example

In this section, we treat an example inspired by studies on a bi-arm servicer, made at MATRA-ESPACE, in the frame of the space shuttle project of the European space station.

A.1 Architecture and Environment

We will describe only the features interesting us, with regard to the mission that we ascribe to the arms.

The components of the world are as follows:

- two arms, *arm1* and *arm2*, initially each at its *home* position, *available* for the execution of a task;
- two cable ends:
  - one with a *male* connector, located at *initposmale*,
  - one with a *female* connector, located at *initposfemale*,

  These two connectors are cylindrical, and made to be mated (i.e. they have the property of being *matable*.)

A.2 The Task: Mating Connectors

We want to use the two arms to mate the connectors, and then place the connected cable in a clip, where it will be fastened.

In order to achieve this :

- *arm1* moves towards *female*'s position, grasps it, and moves to the position *posRVD*.
- *arm2* moves towards *male*'s position, grasps it, and moves to the position *posRVD2*, that is *near posRVD*.

These two subtasks, being independent of each other, are made parallely.

Then, we proceed to the mating of the connectors, itself: these two must first be *aligned*, in order to correct their relative position, then *inserted*, one into the other, then *turned*, so that they are correctly *oriented*, then *pushed*;

finally, the *mate* is *checked* to be fully obtained. As our goal is not, in a first stage, to represent quantitative information, various properties that rely on angles (e.g. *oriented*) or other geometric dimensions (insertion, alignment) are represented here only by a proposition.

After the mating is completed, the connected cable must be fastened, in a cable clip. For that, *arm2*, which is holding the *male* part of the connector, will *ungrasp* it, and then *grasp* the cable, while *arm1* keeps holding the *female* part of the connector.

B Model

At the moment of modeling all of this, we will have to find an expression for the characteristics of the initial state, in terms of temporal facts. A set of simple actions is defined in the formalism of *primitive actions*, corresponding to the actions of the arms, alone or coordinated. Other actions examplify the use of *compound actions*, to define actions having effects dispatched in time, or depending on the context they are executed in. An *elastic action* is used to express the action of keeping doing something.

On these bases, the complete plan for the realization of this task is composed.

B.1 Temporal Facts

B.1.1 Facts

We consider here the initial state, from which the world will evolve under the modifications entailed by the plan.

position(arm1,home(arm1))
position(arm2,home(arm2))
available(arm1)
available(arm2)

position(male,posinitmale)
position(female,posinitfemale)
position(cableclip,poscableclip)

accessible(male)
accessible(female)
accessible(cable)

matable(male,female)
matable(female,male)
matable(cable,cableclip)
near(posRVD,posRVD2)
near(posRVD2,posRVD)
near(poscableclip,connplace)
near(connplace,poscableclip)
near(X,X)

arm(arm1)
arm(arm2)

B.1.2 Rules

We give ourselves some rules, stating that :

- an arm grasping an object *Obj1* mated with another *Obj2*, holds the whole *connector(Obj1,Obj2)* :

  grasping(Arm,connector(Obj1,Obj2)) is true if
  mated(Obj1,Obj2,connector(Obj1,Obj2)) is, and either
  grasping(Arm,Obj1) or *grasping(Arm,Obj2)* is;

- the cable attached to the *female* connector is always placed *near* to it :

  position(cable,P) is true if
  position(female,Posfemale) and *near(Posfemale,P)* are.

- the position of a whole connector, is that of its constituents:

  position(connector(Obj1,Obj2),P) is true if
  position(Obj1,P) or *position(Obj2,P)* is.

B.2 Actions

B.2.1 Primitive and compound actions

Mating and Unmating :

action(align(Obj1,Obj2),
1 ,
[position(Obj1,PosObj1) , position(Obj2,PosObj2) ,
near(PosObj1,PosObj2)] ,
[] ,
[aligned(Obj1,Obj2)]).

action(insert(Obj1,Obj2),
1 ,
[matable(Obj1,Obj2)] ,
[] ,
[inserted(Obj1,Obj2)]).

action(turn(Obj1,Obj2),
1 ,
[aligned(Obj1,Obj2)] ,
[] ,
[oriented(Obj1,Obj2)]).

action(push(Obj1,Obj2),
1 ,
[matable(Obj1,Obj2),aligned(Obj1,Obj2),
inserted(Obj1,Obj2),oriented(Obj1,Obj2)] ,
[inserted(Obj1,Obj2)] ,
[pushed(Obj1,Obj2)]).

action(checkmate(Obj1,Obj2),
1 ,
[pushed(Obj1,Obj2),aligned(Obj1,Obj2),
oriented(Obj1,Obj2)] ,
[pushed(Obj1,Obj2),aligned(Obj1,Obj2),
oriented(Obj1,Obj2)] ,
[mated(Obj1,Obj2,conn(Obj1,Obj2))]).

```
action(  unmate(Obj1,Obj2),
         1 ,
         [mated(Obj1,Obj2,conn(Obj1,Obj2))] ,
         [mated(Obj1,Obj2,conn(Obj1,Obj2))] ,
         ).
```

Grasping and Moving :

- **Grasping/Ungrasping :**
  - Primitive actions :

```
action(  startgrasp(Arm,Obj),
         0 ,
         [available(Arm), accessible(Obj),
            position(Arm,PosArm), position(Obj,PosObj),
            near(PosArm,PosObj)] ,
         [available(Arm), accessible(Obj)] ,
         [busygrasping(Arm,Obj)] ).

action(  dograsp(Arm,Obj),
         1 ,
         [position(Arm,PosArm), position(Obj,PosObj) ,
            near(PosArm,PosObj),
            busygrasping(Arm,Obj)] ,
         [busygrasping(Arm,Obj)] ,
         [grasping(Arm,Obj)] ).
```

  - Dispatched effects (fig. 13) :

```
compound-action(  grasp(Arm,Obj),
                  seq([startgrasp(Arm,Obj),
                     dograsp(Arm,Obj) ])
                  ).
```

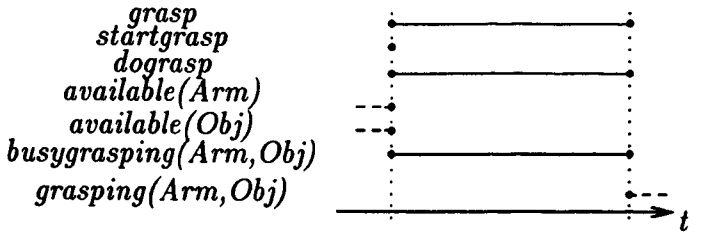


Figure 13: The action *grasp* in time.

– Conditions on the environment :

```
compound-action(  ungrasp(Arm,Obj),
                  cond( mated(Obj1,Obj2,Obj) ,
                     par([ ungraspobj(Arm,Obj),
                        cond(grasping(Arm,Obj1),
                           ungraspobj(Arm,Obj1),
                           ungraspobj(Arm,Obj2) ) ]),
                     ungraspobj(Arm,Obj) )
                  ).
```

with:

```
action(  ungraspobj(Arm,Obj),
         1 ,
         [grasping(Arm,Obj)] ,
         [grasping(Arm,Obj)] ,
         [available(Arm),accessible(Obj) ]
         ).
```

– Another way of putting a piece somewhere :

```
action(  place(Arm,Obj,Place),
         1 ,
         [grasping(Arm,Obj),position(Obj,Place)] ,
         [] ,
         [placed(Obj,Place)] ).
```

- **Moving** :
  - Primitive actions :

```
action(  startmove(Obj,Place),
         0 ,
         [position(Obj,Oldplace)] ,
         [position(Obj,Oldplace)] ,
         [moving(Obj,Oldplace,Place)] ).
action(  domove(Obj,Place),
         1 ,
         [moving(Obj,Oldplace,Place)] ,
         [moving(Obj,Oldplace,Place)] ,
         [position(Obj,Place)] ).
```

  - Dispatched effects :

```
compound-action(  moveobj(Obj,Place),
                  seq([ startmove(Obj,Place),
                     domove(Obj,Place) ])
                  ).
```

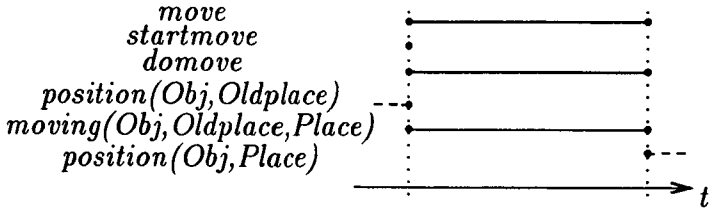


Figure 14: The action *move* in time.

– Conditions on the environment :

```
compound-action(  move(Obj,Place),
                  cond( arm(Obj) ,
                     movearm(Obj,Place),
                     cond( mated(Obj1,Obj2,Obj) ,
                        par([moveobj(Obj1,Place),
                             moveobj(Obj2,Place)]),
                        moveobj(Obj,Place) )
                     )
                  ).
compound-action(  movearm(Arm,Place),
                  cond( grasping(Arm,Obj) ,
                     par([moveobj(Arm,Place),
                        move(Obj,Place)]),
                     moveobj(Arm,Place) )
                  ).
```

Mating, as a compound action :

```
compound-action(  mate(Obj1,Obj2),
                  seq([insert(Obj1,Obj2),turn(Obj1,Obj2),
                     push(Obj1,Obj2),checkmate(Obj1,Obj2)])
                  ).
```

B.2.2 Elastic action

```
elastic-action(  hold(Arm,Obj),
                 [grasping(Arm,Obj)] ,
                 [] ,
                 [] ).
```

LISTE DES DERNIERES PUBLICATIONS INTERNES

PI 480 **THE MODELLING SYSTEM PYRAMIDE AS AN INTERACTIVE HELP FOR THE GUIDANCE OF THE INSPECTION VEHICLE CENTAURE**
Philippe EVEN, Lionel MARCE
22 Pages, Juin 1989.

PI 481 **VERS UNE INTERPRETATION QUALITATIVE DE COMPORTEMENTS CINEMATIQUES DANS LA SCENE A PARTIR DU MOUVEMENT APPARENT**
Edouard FRANCOIS, Patrick BOUTHEMY
40 Pages, Juin 1989.

PI 482 **DEFINITION DE ALPHA : UN LANGAGE POUR LA PROGRAMMATION SYSTOLIQUE**
Christophe MAURAS
18 Pages, Juin 1989.

PI 483 **POLYNOMIAL IDEAL THEORETIC METHODS IN DISCRETE EVENT, AND HYBRID DYNAMICAL SYSTEMS**
Michel LE BORGNE, Albert BENVENISTE, Paul LE GUERNIC,
20 Pages, Juillet 1989.

PI 484 **IMPLEMENTING ATOMIC RENDEZVOUS WITHIN A TRANSACTIONAL FRAMEWORK**
Jean-Pierre BANATRE, Michel BANATRE, Christine MORIN
22 Pages, Juillet 1989.

PI 485 **THE MAPPING OF LINEAR RECURRENCE EQUATIONS ON REGULAR ARRAYS**
Patrice QUINTON, Vincent VAN DONGEN
40 Pages, Juillet 1989.

PI 486 **SYNTHESIS OF A NEW SYSTOLIC ARCHITECTURE FOR THE ALGEBRAIC PATH PROBLEM**
Abdelhamid BENAINI, Patrice QUINTON, Yves ROBERT,
Yannick SAOUTER, Bernard TOURANCHEAU
34 Pages, Juillet 1989.

PI 487 **PLANS SIMULATION USING TEMPORAL LOGICS**
Eric RUTTEN, Lionel MARCE
40 Pages, Juillet 1989.

PI 488 **ON FINITE LOOPS IN LOGIC PROGRAMMING**
Philippe BESNARD
20 Pages, Septembre 1989.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique