Context Dependent Effects in Temporal Planning

Patrick Albers & Malik Ghallab

LAAS-CNRS
7, Av. Colonel Roche, 31077 Toulouse, France
email: {patrick,malik}@laas.fr

Abstract. A major problem in planning, as in most AI domains, is to find an adequate representation. In particular, there is the issue of which effects of an action should be specified unconditionally in its model and which can be stated conditionally with respect to the context. Indeed, most actions do have several different effects depending on the context in which they are executed.
In this paper, we propose an approach and different extensions in order to take into account context dependent effects into IxTeT, a temporal planner. Expressiveness requires a great flexibility of representation, but it may lead to a computational cost not compatible with a practically efficient planner. The proposed approach offers a slight extension in representation which enables to express conditional subtasks. Furthermore, empirical results show that this approach and the corresponding implementation provide also some efficiency benefits with respect to a domain description that details all unconditional action models.

Keywords: temporal planning, actions with context dependent effects

1 Introduction

In planning, a major difficulty is to find an appropriate representation of actions. In particular, the problem is to correctly describe all conditions and effects of an action. Classical planners based on STRIPS-like operators propose an approach where all changes must be specified explicitly. An action may have different effects depending on the context in which it is executed. Consider for example the motion of a robot; if one add the possibility of attaching a trailer to it, a model of the action *move* must consider the position of trailer and all the objects which are on it. Cases like *move without trailer*, *move with attached trailer*, *move with trailer and one object* represent different contexts of the *move* action. If we specify this problem in STRIPS-like operators, we must create as many *move* actions as contexts.

The first non-linear planner to take into account context dependent effects was SIPE [15]. It extends STRIPS representation using deduction operators, which are triggered at each planning step in order to complete the description of the plan. Pednault's Action Description Language (ADL) extends the classical representation [11][12]. Preconditions are partitioned into two sets: primary ones represent the usual preconditions of an action, and secondary ones define the contexts in which an action produces particular effects. The planner *Pedestal* was the first implementation of a linear planner using ADL [10]. It suffered however from a too large algorithmic complexity. The non-linear planner *UCPOP* handles a large subset of ADL; in particular it allows conditional effects [13]. UCPOP is a good compromise between efficiency and expressiveness. Indeed a rich representation is often a conflicting issue with efficiency considerations. One may limit the representation to keep a planner efficient.

This paper focuses on the extension of IXTET a temporal planner. It proposes to generalize its representation in order to take into account context dependent effects. The following section summarizes the IXTET planner. The proposed extension is explained in section 3. Finally, some experimental results showing the benefits of the approach are given.

2 IXTET restricted representation

2.1 Representation

IXTET is based on a reified logic formalism. It uses temporal and atemporal variables. The elementary primitive is the time-point [5]. IXTET handles temporal symbolic constraints (precedence, simultaneity) and numeric constraints expressed by lower and upper bounds on the temporal distance between two points. A time-map manager maintains the consistency of the constraint network and answers queries about time-points.

Atemporal variables ranging over finite domains are managed in a variable constraint network. Propagated constraints are domain restrictions, equalities and inequalities between variable. Inequalities are propagated by a local path consistency algorithm.

The world is described by a set of multi-valued domain attributes, where each attribute is a functional term over atemporal variables. Two predicates are available in IXTET. Predicate $event(att(x_1,...):(v_1,v_2),t)$ says that attribute $att(x_1,...)$ changes its value instantaneously from v_1 to v_2 at time t. Predicate $hold(att(x_1,...):v,(t_1,t_2))$ asserts the persistence of the attribute $att(x_1,...)$ to value v for $t: t_1 \leq t < t_2$. It is used in action models but also by the planner to manage causal links.

IXTET manages resource predicates[8]. There are two types of resources: unsharable ones are single items with a unit capacity (e.g. a key), sharable resources (e.g. electricity) can be used simultaneously by different actions, not exceeding a total maximal capacity. Three operations are possible: $use(typ(r):q,(t_1,t_2))$ borrows a part q of resource between time-points t_1 and t_2; $consume(typ(r):q,t)$ and $produce(typ(r):q,t)$ correspond to consuming and producing a quantity q of a resource at time t.

Planning operators in IxTeT are called tasks. Tasks are *deterministic* operators, without ramification effects. They are composed of: (1) a set of subtasks, (2) a set of events describing the changes of the world induced by the task, (3) a set of assertions (*hold*) to express conditions that should prevail during part of the task, (4) a set of resource uses, and (5) a set of constraints binding the different time-points and atemporal variables in the tasks. The possibility of using subtasks within tasks makes the description hierarchical, but without allowing recursion or other complex control structures. This hierarchy is a programming facility only. A compilation procedure expands subtasks into a flat set of tasks.

The example below describes a planning domain with few constants, attributes and tasks. It is a simple transportation problem with 3 robots, one trailer and several containers. The task *attach_trailer* makes one robot attach a trailer to another distinct robot. The task *put_on_trailer* consists of loading a container onto a trailer, the robot loading the container must not be the one eventually attached to the trailer.

```
constant ROBOTS     = { robot1, robot2, robot3 };
constant TRAILERS   = { trailer1};
constant CONTAINERS = { container1, container2, container3, container4, container5,
container6, container7 };
constant ROOMS     = {Labroom1, Labroom2, Labroom3, Labroom4, Labroom5, Labroom6, RoomD4};

attribute is_attached(?trailer, ?robot)
   {?trailer in TRAILERS; ?robot in ROBOTS; ?value in {yes, no};}
attribute on_trailer(?container, ?trailer)
   {?container in CONTAINERS; ?trailer in TRAILERS; ?value in {yes,no};}

task  put_on_container(?robot, ?trailer, ?container) (start, end) {
  ?robot in ROBOTS; ?trailer in TRAILERS; ?container in CONTAINERS;
variable ?X in ROOMS;
  event( on_trailer(?container,?trailer): (no,yes), end);
  hold ( location(?robot): ?X, (start,end));
  hold ( location(?trailer): ?X, (start,end));
  hold ( location(?container):?X,(start,end));
  event( is_occupied(?robot): (no,yes), start);
  hold ( is_occupied(?robot): yes,(start,end))
  event( is_occupied(?robot): (yes,no), end);
  hold ( is_moving(?trailer): no, (start,end));
  hold ( is_attached(?trailer, ?robot): no, (start,end));
    (end - start) in [ 1:00; 2:00];  }

task attach_trailer(?robot,?robot2,?trailer) (start,end) {
  ?robot in ROBOTS; ?robot2 in ROBOTS; ?trailer in TRAILERS; ?robot2 != ?robot; variable ?X in ROOMS;
  event( is_attached(?trailer,?robot): (no,yes), end);
  hold ( location(?robot): ?X, (start,end))   ; event( is_moving(?trailer): (no,yes), start);
  hold ( location(?robot2): ?X,(start,end))   ; hold ( is_moving(?trailer): yes,(start,end));
  hold ( location(?trailer):?X,(start,end))   ; event( is_moving(?trailer): (yes,no), end);
  event( is_occupied(?robot):(no,yes), start) ; event( is_occupied(?robot2): (no,yes),start);
  hold ( is_occupied(?robot): yes,(start,end)); hold ( is_occupied(?robot2): yes,(start,end));
  event( is_occupied(?robot):(yes,no), end)   ; event( is_occupied(?robot2): (yes,no), end);
    (end - start) = 0:30; }
```

In IxTeT, preconditions and effects do not appear explicitly separated within a task, since the same event predicate can express both for its two values. The temporal representation enables to easily express the various things that hold before, what prevail during which part of the action, or that change as an effect of the action and when this change takes place along the duration of the action.

A simple analysis procedure of task models provides at compilation time the separation of preconditions from effects.

2.2 The planner

IxTeT relies on the closed world hypothesis. The initial plan $\mathcal{P}_{init}$ describes the problem scenario, that is: (1) initial values for all instantiated attributes, (2) expected availability profile of resources, and (3) goals that must be achieved. The user has the possibility of describing a dynamic domain with contingent events [6], *i.e.* what is expected to happen in the world while the plan will be executed. This description is assumed to be complete; events on the same instantiated attribute should be totally ordered with compatible values.

Starting with the initial plan $\mathcal{P}_{init}$, the search incrementally constructs a solution by refining a partial plan through the insertion of tasks or variable constraints. IxTeT is based on the classical notion of *causal links* [9][14], handled in our case by the*hold* predicate. The search control relies on an *extended least commitment* strategy according to a near admissible algorithm A_ϵ. IxTeT also uses a dynamic hierarchy of domain attributes which guides the choice of flaws to be solved [4], a *flaw* being either an unexplained proposition, a threat or a resource conflict.

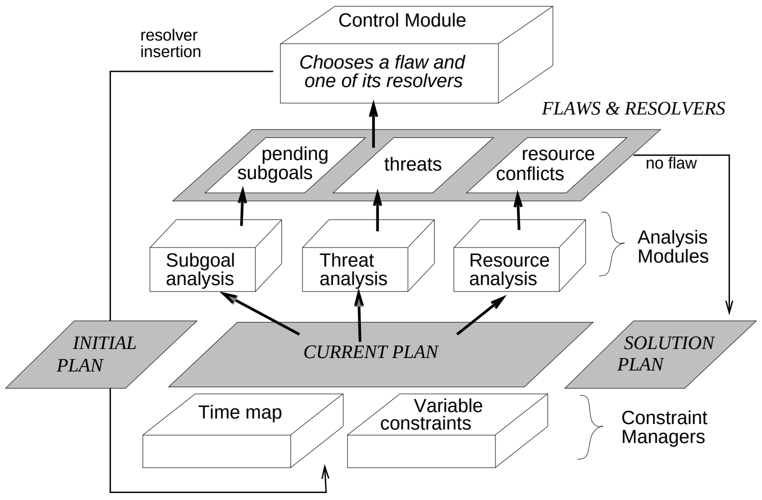

Fig. 1. the IxTeT planner architecture

A partial plan $\mathcal{P}$ is a solution when all goals and sub-goals are satisfied, and there are no causal link or resource conflicts. The global IxTeT architecture is presented in figure 1, and detailed in [6][8].

3 IxTeT with context dependent effects

The proposed extension remains within the usual assumptions of a closed world and deterministic actions. Contexts and corresponding conditional effects must be defined explicitly within a task model. A task may contain several conditional parts, each corresponding to a particular context, let call these *conditional subtasks*. When a task is added to a partial plan, it is not always possible to know if a context holds or not. Hence, all conditional parts of a new task remain hypothetical and need to be kept as such. Consequently, a partial plan is split into two different parts. The first one is composed of the unconditional propositions of the partial plan (*i.e.* events, assertions, resource uses and variable constraints). The second part corresponds to the hypothetical conditional subtasks of the partial plan.

Since the conditional parts of a partial plan are hypothetical, only non-conditional propositions (event and hold) can contribute to open subgoals. On the other hand, all effects of a partial plan, *i.e.* conditional or not, may be a threat to a causal link [9]. Similarly, resource analysis must handle all resource uses, conditional or not.

Managing context-dependent effects does not change the global architecture of IxTeT (figure 1). However, each analysis module needs be modified. Let us first present the representation and define partial plans with hypothetical conditional parts before explaining these modifications.

3.1 Conditional subtasks

A conditional subtask may contain all that can be expressed within a task. It has two field : **if** (*condition-field*) { *body-field* }

An element of the condition-field may be either an assertion (hold) or an atemporal constraint. An element of the body-field may contain any proposition. For example, the task below specifies additional effects for a robot motion attached to a trailer:

```
{task move(?robot,?from, ?to) (start, end)
  {?robot in ROBOTS; ?from in PLACES; ?to in PLACES; variable ?trailer in TRAILERS;
  event(location(?robot):(?from,Inway),start);
  hold (location(?robot): Inway, (start,end));
  event(location(?robot):(Inway,?to), end);

  if( hold(is_attached(?trailer,?robot): yes, (start,end))) {
      event(location(?trailer):(?from,Inway), start);
      hold (location(?trailer): Inway, (start,end));
      event(location(?trailer):(Inway,?to), end);    }
  ?from != ?to;  (end - start) in [03:00, 5:00];   }}
```

Since in IxTeT preconditions are not explicitly separated from effects, the body-field a conditional subtask, relying on the same temporal representation, may also contain implicit preconditions. The condition-field enhances some preconditions in order to simplify programming. For example, in the task *move*, the conditional subtask has two preconditions: $is\_attached(?trailer, ?robot) = yes$ during the action, and $location(?trailer) = location(?robot)$ before time $start$.

An effect in a conditional subtask may be unwanted in a plan. A solution to such a conflict is to make sure that the conditional subtask does not apply; *i.e.* one of its preconditions does not hold. Possible resolvers for such a conflict are *negations* of preconditions of a conditional subtask.

For that, let us focus on the conditions in the conditional-field, without taking into account other preconditions eventually in its body-field. This restriction limits the number of resolvers. The condition-field of a conditional subtask is composed of assertions and atemporal constraints. The negation of a constraint is its complement. The negation of an assertion: $hold(att(x_1, ..., x_n) : v, (t_1, t_2))$ states that this attribute takes another value during interval $[t_1, t_2[$. This is expressed by the assertion $hold(att(x_1, ..., x_2) : v', (t_3, t_4))$ with the constraints: $v \neq v'$, and $t_1 < t_3 < t_4 < t_2$.

Note that by restricting the set of resolvers to the condition-field only, we lose the search completeness. However, one may choose to specify all preconditions within the condition-field.

3.2 Partial plans with conditional subtasks

Let us assume here that a conditional subtask cannot contain other conditional subtasks. We'll handle later nested conditionals.

Let $\mathcal{P}$ be a partial plan: $\mathcal{P} = (EVT, HLD, RES, IF, CV, CT)$, where EVT is the set of event predicates, HLD the set of assertions, RES the set of resources, IF the set of elements in the condition-field of all conditional subtasks in $\mathcal{P}$, CV and CT represent the sets of constraints on atemporal and temporal variables respectively.

Each set EVT, HLD, USE, CV and CT is decomposed into two subsets: a set of conditional propositions (which come from a conditional subtask), noted with subscript cd, and a set of non-conditional propositions with subscript nc. The set IF is decomposed into two sets: the set IF_h contains assertions, and IF_v which contains atemporal constraints. A partial plan is then defined in the following way:

$$\mathcal{P} = ((EVT_{nc}, EVT_{cd}), (HLD_{nc}, HLD_{cd}), (RES_{nc}, RES_{cd}),$$
$$(IF_h, IF_v), (CV_{nc}, CV_{nc}), (CT_{nc}, CT_{cd}))$$

The conditional sets contain all conditional subtasks which are still hypothetical. In order to differentiate between these, an index is associated to the elements of a conditional subtask.

A conditional subtask in a partial plan may either:

1. have an effect which explains a subgoal. In this case, the conditional subtask may be needed, it can be transfered from the conditional sets to the non-conditional sets. In that case, all preconditions of this conditional subtask contribute to open subgoals.

2. have an effect which may be a threat to a causal link. A resolver for this flaw, *i.e.* the negation of one condition, is added to the partial plan. The conditional subtask is removed from the partial plan, since it cannot be later verified[1].
3. or none of the two preceding cases holds, the conditional subtask remains hypothetically in the plan.

Let $\mathcal{C} = (\overline{EVT}, \overline{HLD}, \overline{RES}, \overline{IF_h}, \overline{IF_v}, \overline{CV}, \overline{CT})$ be a conditional subtask of index i, where the sets $\overline{EVT}$, $\overline{HLD}$, $\overline{RES}$, $\overline{CV}$, $\overline{CT}$ specify the body-field of the conditional subtask; the sets $\overline{IF_h}$ and $\overline{IF_v}$ represent its condition-field.

Formally these sets are:

$$\begin{aligned}
\overline{EVT} &= \{event | event \in EVT_{cd} \wedge index(event) = i\} \\
\overline{HLD} &= \{hold | hold \in HLD_{cd} \wedge index(hold) = i\} \\
\overline{RES} &= \{res | res \in RES_{cd} \wedge index(res) = i\} \\
\overline{IF_h} &= \{hold | hold \in IF_h \wedge index(hold) = i\} \\
\overline{IF_v} &= \{cv | cv \in IF_v \wedge index(cv) = i\} \\
\overline{CV} &= \{cv | cv \in CV_{cd} \wedge index(cv) = i\} \\
\overline{CT} &= \{ct | ct \in CT_{cd} \wedge index(ct) = i\}
\end{aligned}$$

The insertion of a conditional subtask $\mathcal{C}$ into a partial plan $\mathcal{P}$ (case 1 above) is performed as follows:

$$\begin{gathered}
\mathcal{P} \oplus insertion(\mathcal{C}) = \\
((EVT_{nc} \cup \overline{EVT}, EVT_{cd} - \overline{EVT}), (HLD_{nc} \cup \overline{HLD} \cup \overline{IF_h}, HLD_{cd} - \overline{HLD}), \\
(RES_{nc} \cup \overline{RES}, RES_{cd} - \overline{RES}), (IF_h - \overline{IF_h}, IF_v - \overline{IF_v}), \\
(CV_{nc} \cup \overline{CV} \cup \overline{IF_v}, CV_{cd} - \overline{CV}), (CT_{nc} \cup \overline{CT} CT_{cd} - \overline{CT}))
\end{gathered}$$

The insertion of a resolver for a conflict with a conditional subtask $\mathcal{C}$ (case 2) corresponds to:

$$\begin{gathered}
\mathcal{P} \oplus negation(\mathcal{C}) = \\
\bigvee_{hold \in \overline{IF_h}} ((EVT_{nc}, EVT_{cd} - \overline{EVT}), (HLD_{nc} \cup \{\neg(hold)\}, HLD_{cd} - \overline{HLD}), \\
(RES_{nc}, RES_{cd} - \overline{RES}), (IF_h - \overline{IF_h}, IF_v - \overline{IF_v}), \\
(CV_{nc}, CV_{cd} - \overline{CV}), (CT_{nc}, CT_{cd} - \overline{CT})) \\
\bigvee_{cv \in \overline{IF_v}} ((EVT_{nc}, EVT_{cd} - \overline{EVT}), (HLD_{nc}, HLD_{cd} - \overline{HLD}), \\
(RES_{nc}, RES_{cd} - \overline{RES}), (IF_h - \overline{IF_h}, IF_v - \overline{IF_v}), \\
(CV_{nc} \cup \{\neg(cv)\}, CV_{cd} - \overline{CV}), (CT_{nc}, CT_{cd} - \overline{CT}))
\end{gathered}$$

We remove the conditional subtask from the partial plan, and add the negation of one of its conditions. Note that the choice of a resolver and possible backtrack points are handled by the control module. Again, possible preconditions which are in the body-field of the conditional subtask are not taken into account.

3.3 Search control

Subgoal analysis Subgoals are expressed in IxTeT by unexplained propositions. Explained propositions are events or assertions which are established by a causal

[1] Weld calls this *confrontation* [14].

link. In IxTeT, establishers are naturally represented by events and causal links by assertions. To explain a proposition, the planner must find an event either already in the partial plan or in a task model that needs to be added to it.

Let $\mathcal{E}$ be an event which may establish an unexplained proposition P of partial plan $\mathcal{P}$. The procedure for handling P is the following:

1. If $\mathcal{E}$ belongs to $\mathcal{P}$ and is not conditional: add a causal link between $\mathcal{E}$ and P, and corresponding variable binding constraints
2. If $\mathcal{E}$ belongs to a conditional subtask $\mathcal{C}$ of $\mathcal{P}$: add the conditional subtask $\mathcal{C}$ which contains $\mathcal{E}$, and add a causal link between $\mathcal{E}$ and P.
3. If $\mathcal{E}$ belongs to a task to be added to $\mathcal{P}$ and $\mathcal{C}$ is not conditional: add a new instantiated task and a causal link between $\mathcal{E}$ and P.
4. If $\mathcal{E}$ belongs to a conditional subtask $\mathcal{C}$ of a task to be added: add the new instantiated task, the conditional subtask $\mathcal{C}$, and a causal link between $\mathcal{E}$ and P.

Resolvers of types (2) and (4) have been added in IxTeT in order to take into account conditional subtasks.

Threat analysis An assertion $\mathcal{H} : hold(att(x_1, ..., x_n) : v, (t_1, t_2))$ of a partial plan is threatened if an event $\mathcal{E} : event(att(x'_1, ..., x'_n) : (v'_1, v'_2), t')$ or an assertion $\mathcal{G} : hold(att(x_1", ..., x_n") : v", (t_1", t_2"))$ concerning the same attribute happens at overlapping intervals. Propositions $\mathcal{H}$, $\mathcal{G}$ or $\mathcal{E}$ may or may not belong to a conditional subtask.

A threat is a couple:

1. $< \mathcal{H}, \mathcal{G} >$ such that the partial plan does not contain one of those constraints: $\{(t_2" < t_1); (t_2 < t_1"); (x_1 \neq x_1"); ...; (x_n \neq x_n"); (x_1 = x_1" \wedge ... \wedge x_n = x_n" \wedge v = v")\}$, or
2. $< \mathcal{H}, \mathcal{E} >$ such that the partial plan does not contain one of those constraints: $\{(t' < t_1); (t_2 < t'_1); (x_1 \neq x_1"); ...; (x_n \neq x_n"); (t' = t_1 \wedge v'_2 = v); (t' = t_2 \wedge v'_1 = v)\}$.

For each possible threat, a disjunction of the above temporal and atemporal constraints is found. If the threatening proposition ($\mathcal{G}$ or $\mathcal{E}$) belongs to a conditional subtask, this subtask cannot stay hypothetical. We may either add it into the partial plan, together with one of the above constraints to avoid the threat, or we may remove it, but add the negation of one of its conditions.

Resource analysis There is a conflict for a type *typ* of resource with a maximal capacity Q, if there exists a set $U = \{u_1, ..., u_n\}$ with $u_i : use(typ() : q^i, (t^i_1, t^i_2))$, such that: (1) $\Sigma^n_{i=1} q_i > Q$, and (2) the partial plan does not contain one of the constraints $\{(t^i_2 < t^j_1)\}_{(i \neq j)}$.

Each u_i may or may not belong to a conditional subtask.

Resolvers of a resource conflict consist of a disjunction of temporal constraints, variable inequalities if different resources of the same type are allowed, and if u_i belongs to a conditional subtask, the negations of its conditions. A resource conflict may also be solved by adding a task producing this resource, if possible.

Solution plan criterion A *flaw* being either an unexplained proposition, a threat or a resource conflict, a partial plan is a solution if and only if: (1) there are no flaws, (2) the temporal constraint network is consistent, and (3) the variable binding network is consistent.

A conditional subtask introduced by a task while planning may remain in the set of conditional subtasks according to the least commitment strategy, unless required for explaining a subgoal or solving a threat. However, a conditional subtask may be verified if all its preconditions are true in a final plan, *i.e.* all its conditions as well as all the preconditions which appear in its body do hold. In this case the conditional subtask must be included unconditionally in the plan. If we do not complete a plan in this way, it will remain consistent; but the predicted world states will not be correct: additional effects have to be added. Notice that the completion of a final plan with respect to its conditional subtasks that remains hypothetical, can be pursued during plan execution time. This can be helpful for monitoring purposes.

4 Illustration and results

4.1 A detailed example

As described in section 2.1, the chosen example concerns three robots, one trailer and seven containers. The trailer cannot be attached to two different robots. A robot cannot pick up or put on a container on a trailer, when the trailer is moving, or when the trailer is attached to this robot.

Initially the 3 robots, the trailer and all the containers are in $RoomD4$. Containers number 3, 4 and 5 are on the trailer. The goal is to keep $container3$ in $RoomD4$ during the totality of the plan; containers 1 and 2 must be in $LabRoom1$, and $containe4$ in $LabRoom2$. The solution plan is presented in figure 2; the solution plan is found in 393 steps (number of developed partial plans) and with four backtracks.

Let us analyze an early partial plan where the goal *"*$container3$ *stays in* $RoomD4$*"* is satisfied by a causal link between this goal and the event in the initial situation; the goal *"*$container1$ *in* $LabRoom1$*"* is satisfied by an instantiated task *move* with the appropriate conditional subtask.

The subgoal analysis module could satisfy the goal $container2$ *in* $LabRoom1$ in two ways: add the conditional subtask of the task *move*, or add a new instantiated task *move*. Adding a new instantiated task *move* satisfies as well the goal $container4$ *in* $Labroom2$. The threat analysis module detects that the conditional subtask of the task *move* in this partial plan, which concerns $container3$, threatens the previously established causal link. The only possible resolver for

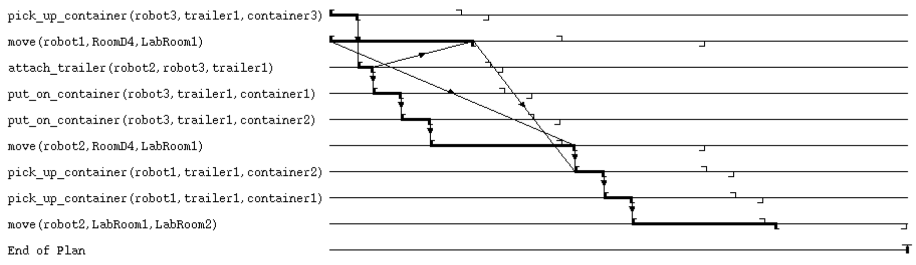
pick_up_container(robot3,trailer1,container3)
move(robot1,RoomD4,LabRoom1)
attach_trailer(robot2,robot3,trailer1)
put_on_container(robot3,trailer1,container1)
put_on_container(robot3,trailer1,container2)
move(robot2,RoomD4,LabRoom1)
pick_up_container(robot1,trailer1,container2)
pick_up_container(robot1,trailer1,container1)
move(robot2,LabRoom1,LabRoom2)
End of Plan

Fig. 2. Solution plan

this flaw is to insert the negation of this conditional subtask (*i.e. container*3 must not be on the trailer). There are other threats to temporal or atemporal constraints. The heuristically chosen flaw and resolver is the negation of the conditional subtask, because this flaw has just one resolver.

When a solution plan is found, the completion phase detects that the position of *container*5, which is on the trailer at the beginning of the plan, changes. We can see here that the completion is a necessary step; otherwise, the solution plan will not say that *container*5 has moved along with the trailer and the robot.

4.2 Some results

Several domains and many problems have been tested with similar results. In the multi-robots transportation domain for example, there are 129 different possible contexts for the *move* action, but only 8 conditional subtasks (one trailer and seven containers).

We tried to compare the system extended with conditional subtasks to the standard approach of IxTeT, *i.e.* without conditional subtasks. The task *move* must be rewritten differently as many times as there are contexts. Some representative results are presented below; the 4 examples correspond to different goals and initial situations:

	example 1	example 2	example 3	example 4
without conditional subtasks	35/0/15.1	86/0/67.8	492/5/408.6	415/4/522.4
with conditional subtasks	29/0/2.6	76/0/41.2	391/8/279.6	58/0/8.6

(number of developed partial plans / number of backtracks / time in seconds)

The results depend on the problem to be solved and the number of *move* tasks required. The threat module requires more time with conditional subtasks (about 80% more). Indeed, for each insertion of a task *move*, all contexts must be taken into account. However, without conditional subtasks the planner spends more time in the subgoal analysis; the average branching factor is larger.

Our experiments reports that generally IxTeT with context dependent effects is faster. But this does not always hold. It depends on the position of the solution in the search tree: the branching factor being smaller, there are fewer backtracks. This is may be a significant advantage. In particular, if there is no solution, planning with conditional subtasks takes less time. But there is an overhead, mainly due to the threat analysis module.

In order to estimate this overhead as a function of the number of conditional subtasks, we used the same simple domain varying the numbers of containers and hence the number of conditional subtasks; notice that this does not change the planning. The results are presented in the following table.

nb of containers	1	7	14	21	28	35	42	49
planning time in seconds	2	2.9	4.2	5	5.9	7.7	9.2	10.6

It seems that the increase in time depends linearly on the number of conditional subtasks.

5 Conclusion

Temporal planning with context dependent effects is interesting for two essential reasons: it improves the expressiveness of the representation and it reduces the branching factor. Altogether, it seems to have a benefit in efficiency, even if the complexity of the threat analysis module increases linearly with the number of conditional subtasks.

The fact of characterizing a context through a subset of preconditions in a conditional subtask may improve the efficiency of the search, but this is paid by a significant drawback: the loss of completeness of the planner. Again, a programming flexibility remains here which enables to overcome this drawback.

The case of nested conditionals is easily handled using partially ordered indexes, such that if $i < j$, then the conditional subtask i contains the subtask j. When a subtask j is added, all subtasks i, such that $i < j$, must be added too in a partial plan. When a conditional subtask j is removed, all subtasks i such as $j < i$ must be removed.

Finally, it is possible to reduce the number of conditional subtasks in a partial plan: when a precondition of a conditional subtask does not necessarily hold; *i.e.* its negation is in the partial plan, this subtask can be removed. According to our preliminary experiments, this reduces the time of the threat analysis module sufficiently to pay for the extra work involved to detect those situations.

References

1. D. Chapman. *Planning for Conjunctive goals.* Artificial Intelligence, vol. 32, pages 333–377, 1987.
2. G. Collins & L. Pryor, *Achieving the functionality of filter conditions in a partial order planner.* AAAI 92.
3. R.E.Fikes & N.J.Nilsson. *STRIPS: a new approach to the application of theorem proving to problem solving.* Artificial Intelligence, vol 2, pages 189–208, 1971.
4. F. Garcia & P. Laborie, *Hierarchisation of the search space in temporal planning.* 235-249, proceedings EWSP-95.

5. M. Ghallab & T. Vidal, *Focusing on the sub-graph for managing efficiently numerical temporal constraints.* In proceedings, FLAIRS-95.
6. M. Ghallab & H. Laruelle. *Representation and control in IxTeT, a temporal planner.* In proceedings AIPS-94, 61-67.
7. S. Kambhampati, C. Knoblock, Q. Yang. *Planning as rafinement search; a unified framework for evualuting tradeoffs in partial order planning.* Artifical Intelligence, 95, 167-238.
8. P. Laborie & M. Ghallab. *Planning with sharable resource constraints.* In proceedings IJCAI-95, 1643-1649
9. D. McAllester & D. Rosenblitt. *Systematic nonlinear planning.* In proceedings AAAI-91, pages 634-639, 1991.
10. D. McDermot, *regression planning.* International Journal of Intelligent Systems 6:357-416, 1991.
11. E. Pednault. *Synthesizing plans that contains actions with context-dependent effects.* Computational Intelligence, 4(4):356-372, 1988.
12. E. Pednault. *Generalizing non-linear planning to handle complex goals and actions with context-dependents effects.* Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, july 1991.
13. J.S.Penberty & D. Weld. *UCPOP: a sound, complete, partial planner order planner for ADL.* Third International Conference of Knowledge Representation and Reasoning, 103–114, october 1992.
14. D. Weld *An introduction to least-commitment planning.* AI magazine, pages 27-61, Winter 1994
15. D.E. Wilkins, *Representation in a domain independent planner.* IJCAI 83