Planning with Inaccurate Temporal Rules

Mathieu Guillame-Bert
INRIA Rhône-Alpes Research Center
655 Av. de l'Europe, 38330 Montbonnot-St.-Martin
France
Email: mathieu.guillame-bert@inrialpes.fr

James L. Crowley
INRIA Rhône-Alpes Research Center
655 Av. de l'Europe, 38330 Montbonnot-St.-Martin
France
Email: james.crowley@inrialpes.fr

***Abstract*—We use a temporal pattern model called Temporal Interval Tree Associative Rules (Tita rules). This pattern model has been introduced in a previous work. The model can express uncertainty, temporal inaccuracy, the usual time point operators, synchronicity, incomplete orders, chaining, disjunctive time constraints and temporal negation. This pattern model is initially designed to be used for temporal learning. In this paper, we use Tita rules as world description models for a Planning and Scheduling task. We present an efficient temporal planning algorithm able to deal with uncertainty, temporal inaccuracy, discontinuous (or disjunctive) time constraints and predictable but imprecisely time located exogenous events. We evaluate our technique by joining a learning algorithm and our planning algorithm into a simple reactive cognitive architecture that we apply on with virtual robot.**

***Keywords*-uncertainty; inaccuracy; disjunctive temporal constraints; automated planning and scheduling; symbolic time sequences; robotic cognitive architecture**

I. Introduction

Automated Planning and Scheduling is the study of computer programs designed to building strategies that lead to desired states. Domains of application include autonomous robotics, automatic system maintenance and project management. The input of a temporal planning and scheduling algorithm is a description of a 'world', a list of possible doable actions, and a desired goal (or set of desired goals). The term *world* is a generic term that represents an initial state and the model of a system (this model is often presented a set of rules). The output of a Planning algorithm is a plan, i.e. a description of the actions to perform in order to achieve a goal.

In [1], we introduce a pattern model for temporal associative rules called Temporal Interval Tree Association Rule (or Tita rule). Tita rules can express both uncertainty and temporal inaccuracy. They can also express the usual time point operators, synchronicity, incomplete ordering, chaining disjunctive time constraints and temporal negation. The Titarl algorithm (Temporal Interval Tree Associative Rule Learner) is a temporal learning algorithm able to efficiently extract Tita rules from symbolic time sequences (sequence of time-sampled symbols).

The goal of this research is to use Tita rules as world description models in a Planning and Scheduling task. Since Tita rules can represent imprecise (non-deterministic) and inaccurate temporal relations, the Planning and Scheduling technique we are introducing has to be able to deal with such aspects. The tree structure of Tita rules (by opposition to the complete graph structure of usual pattern models) allows efficient planning with disjunctive time constraints and inaccurate temporal relations. Time is considered to be continuous.

The first stage of the algorithm, called Titar planner, is able to build a plan from a goal and a single rule. The grammar of the plan we develop is called a Titar Plan. The created plans have a certain amount of freedom and do not necessarily impose an exact time for actions to be performed: For example, if the achievement of an action at any time in a given period leads to a goal, a good plan would define this period instead of assigning an arbitrary time-stamp to the actions.

Since a world is generally not described with one single rule, we design an algorithm able to combine rules to build a plan. However, because of the temporal inaccuracy of rules, given a goal and a chain of high confidence rules leading to this goal, a direct plan (a Titar plan) might not always have high confidence. This point is discussed in details in the next section. In order to solve this problem, we propose a plan model called Meta Titar plan. Informally, a Meta Titar plan is a set of Titars plans connected by temporal constraints. A Meta plan is more expressive than a plan. A Meta plan can express concepts such as 'waiting on an expected exogenous event before continuing the execution of a sequence of actions'. We present an algorithm able to build Meta plans from a goal, a set of rules, and eventually additional observations of the world. The last part of the algorithm is the scheduling stage. The scheduling stage is the selection of the time samples of actions to perform according to arbitrary criteria.

The next section is a quick review of the related literature. The third section defines the Tita rules pattern model and the Titar plan pattern model. Simple examples are presented. The fourth section introduces the planning and scheduling algorithm we have developed. The fifth section presents an example of the use of our planning and scheduling algorithm. In this example, we combine our learning algorithm with our planning algorithm to form a simple cognitive architecture. The cognitive architecture is used to control a robot in a virtual world. We discuss several aspects of the algorithm.

II. Related work

Classical planning problems can be solved by forward chaining [2], backward chaining [3], SAT reduction [4],

model checking, heuristics, among other techniques. Temporal Planning and Scheduling extends classical planning with a temporal aspect. The temporal aspect allows non zero-duration and overlapping actions, and inaccurately located events.

The *Temporal Constraint Satisfaction Problem* (TCSP) is the problem of determining if the time-samples of a given Temporal Constraint Network [5] (or TCN) can be assigned in such way as to make all its constrains valid. TCSP is intractable in the general case (problem of disjunction). Solving STP (TCN where constraints are restricted to be intervals) is a polynomial problem. Balaban et al. [6] define a sub-class of TCN for the TCSP. This class defines the TCN that can be solved with a divide and conquer strategy.

Deviser [7] is a planning and scheduling algorithm that solves goals such as "Make X true between t_1 and t_2 for at least duration n". The algorithm needs a deterministic (non-probabilistic) description of the world (states and rules). The algorithm allows non-zero duration events. The planning stage of the algorithm produces a plan (represented as PERT chart).

Planning on Disjunctive Temporal Constraints is an NP-hard problem. Algorithms such as DT-POP [8] propose heuristic based strategies to deal with this problem.

The world is generally considered to be partially uncertain. Algorithms able to do planning with uncertainty have been proposed. STPU [9] consider the Temporal Constraint Satisfaction Problem where some nodes' time-stamps are unknown and cannot be specified. In the case of STPU, constraints are restricted to be non disjunctive.

III. Tita rules and (Meta) Titar plans

A. Basics

A *probability distribution* describes the probability of each value (or interval of values) of a random variable. The uniform probability distribution between two points a and b is noted $\mathbb{U}_{a,b}$.

Suppose a probability distribution of a continuous variable $f : \mathbb{R} \to \mathbb{R}^+$. By convention, the probability distribution $f' := f+x$ with $x \in \mathbb{R}$ is defined as $f' : t \mapsto f(t-x)$. This operation can be interpreted as a translation of the distribution.

A *(temporal) event* e is a symbol (called type and noted symbol_e) and a time of occurrence (time_e).

A *state* s is a function $\mathbb{R} \to \{0, 1\}$ that maps a value for every time location (i.e. real number). If $s(t) = 1$, s is said to be true at time t. Otherwise, s is said to be false at time t.

A *Boolean function* is a function $\mathbb{R} \to \{0, 1\}$. In this work, Boolean functions are used to represent sub-sets of $\mathbb{R}$: A Boolean function b represents the set $\{x | b(x) = 1\}$.

We define the Boolean function $\mathbb{B}_{a,b}$ as:

$$\mathbb{B}_{a,b} : x \mapsto \begin{cases} 1 & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A *convolution* is a mathematical operation on two functions, producing a third function. The convolution of f and g is written $f * g$ and is defined as follow:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x - t)g(t)dt \quad (2)$$

Convolution of two Boolean functions is a *piecewise continuous linear function* which can be computed very efficiently by an appropriate algorithm. Convolution between Boolean functions is equivalent to Dechter's composition $\otimes$ operator [5]. Convolution between a Boolean function and a probability distribution extends the Dechter's composition operator to probabilistic time distribution.

B. Tita rules

A Temporal Interval Tree Associative Rule (or Tita rule) is a temporal pattern with the semantic of a rule (i.e. a body and a head) and the structure of a tree of temporal constraints. Several graphical examples of rules are given in fig. 1 and detailed after the formal definition.

A *type 1 condition* c is a symbol (symbol_c) and a set (possibly empty) of type 2 conditions (conds_c). The writing convention is $c := \langle \mathrm{symbol}_c, \mathrm{conds}_c \rangle$. Given a set of events E, a type 1 condition c is true at time t if:

- E contains an event e of symbol symbol_c and time t i.e. $\mathrm{symbol}_e = \mathrm{symbol}_c$ and $\mathrm{time}_e = t$.
- All type 2 conditions $c' \in \mathrm{conds}_c$ are true at time t (see definition bellow).

A *type 2 condition* c is either:

- The negation of a type 2 condition c_2 (written $c := \mathrm{not}\, c_2$). Here, c is true at time t if and only if c_2 is false at time t.
- A condition over a state s (written $c := s$). Here, c is true at time t if and only if s is true at time t i.e. $s(t) = 1$.
- An association between a Boolean function m and a type 1 condition c_3 (written $c := [m, c_3]$). Here, c is true at time t if and only if $\exists t'$ with $m(t' - t) = 1$ and c_3 is true at time t'. The Boolean function is the temporal constraint of the condition.

A *Temporal Interval Tree Associative Rule* (Titar) r is a symbol (symbol_r), a confidence (conf_r), a non null temporal distribution (dist_r) called the head, and a type 1 condition (cond_r) called the body. $\mathrm{dist}_r(t - t')$ is the probability density of having an event of symbol symbol_r at time t while the condition cond_r being true at time t'. The temporal distribution is the temporal constraint of the rule's head. The writing convention is $r := \mathrm{cond}_r \Rightarrow \mathrm{head}_r \langle \mathrm{conf}_r, \mathrm{dist}_r \rangle$.

When the body cond_r of a Tita rule r is true at time t (also written as $\mathrm{cond}_r(t)$), r is said to predict an event of symbol head_r with a probability of conf_r and with a temporal distribution of $t + \mathrm{dist}_r$. An event e of symbol $\mathrm{symbol}_e = \mathrm{head}_r$ is said to verify such prediction if the density of the prediction is not equal to zero at time time_e i.e. $f'(\mathrm{time}_e) > 0$ with $f' := t + \mathrm{dist}_r$. By convention, the *temporal precision* of a rule r is defined as $\frac{1}{\mathrm{range}(dist_r)}$.

We present four examples of Tita rules. The graphical representation of the rules is given in fig. 1. This representation is made to help the understanding and reading of rules.

Example III.1. *Suppose the rule* $r_1 := \langle A, \emptyset \rangle \Rightarrow B \langle 95\%, \mathbb{U}_{10,15} \rangle$. *Literally,* r_1 *expresses that if an event of type* A *occurs at time* t*, then, an event of type* B *will occur between*

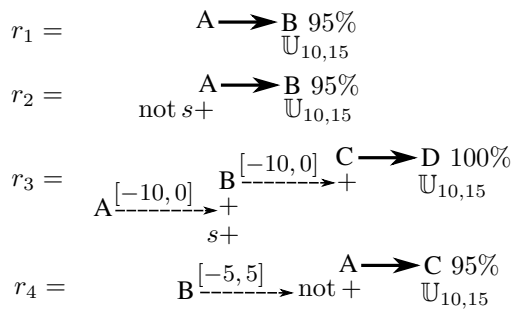

Fig. 1. Four examples of Titar rules

$t+10$ *and* $t+15$ *with* 95% *chance. This rule is a* unit rule*. It expresses a simple and direct correlation between two events.*

Example III.2. *Suppose the rule* $r_2 := \langle A, \{\text{not } s\}\rangle \Rightarrow B\,\langle 95\%, \mathbb{U}_{10,15}\rangle$*. Literally,* r_2 *expresses that if an event of type* A *occurs at time* t *and the state* s *is not true at time* t*, then, an event of type* B *will occur between* $t+10$ *and* $t+15$ *with* 95% *chance. This rule uses the negation of a state as a condition.*

Example III.3. *Suppose the rule* $r_3 := \langle C, \{[T_{-10,0}, \langle B, \{s, [T_{-10,0}, \{\langle A, \emptyset\rangle\}]\}\rangle]\}\rangle \Rightarrow D\,\langle 100\%, \mathbb{U}_{10,15}\rangle$ *. Literally,* r_3 *expresses that if an event of type* C *occurs at time* t_c *followed by an event of type* B *at time* t_b *(with a maximum interval of 10 seconds i.e.* $t_b - 10 \leq t_c \leq t_b - 0$*) followed by an event of type* A *at time* t_a *(with a maximum interval of 10 seconds i.e.* $t_a - 10 \leq t_b \leq t_a - 0$*) and* s *is true at time* t_b*, then, an event of type* D *will occur between* $t_a + 10$ *and* $t_a + 15$ *with* 100% *chance. This rule is the chain of conditions* $C \to B \to A \overset{then}{\Longrightarrow} D$*.*

Example III.4. *Suppose the rule* $r_4 := \langle A, \{\text{not } [T_{-5,5}, \langle B, \emptyset\rangle]\}\rangle \Rightarrow C\,\langle 95\%, \mathbb{U}_{10,15}\rangle$*. Literally,* r_4 *expresses that if an event of type* A *occurs at time* t *and no events of type* B *occur between* $t-5$ *and* $t+5$ *i.e. there are no events of type* B *around the event of type* A*, then, an event of type* C *will occur between* $t+10$ *and* $t+15$ *with* 95% *chance. This rule shows a negation of the occurrence of an event.*

C. Titar plans

We define the notion of Titar plan and give some examples. We consider two types of symbols: The *doable symbols* are symbols of events that be triggered directly, and the *exogenous symbols* are symbols of events that can only be observed. Every doable symbol is associated to a Boolean function describing when it is doable. In the case of on-line planning, this constraint is a way to specify that we can't trigger actions in the past.

Definition III.5. *A* Titar plan *is a directed forest (disjoint union of tree graphs) expressing a set of constraints over temporal events. The edges are oriented from the trunks to the leaves. Every vertex* v *is labelled with a tuple containing a symbol (*symbol_v*), a Boolean function* (Acst_v) *and a set of state symbols (*states*). The Boolean function of a vertex is called the* absolute constraint*. Every edge* e *is labelled with a Boolean function noted* Rcst_e*, and a label 'positive' or 'negative'. The Boolean function of an edge is called the relative constraint.*

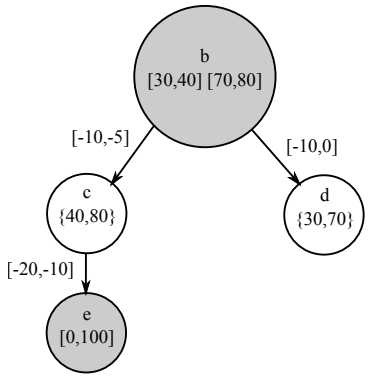

Fig. 2. Example of Titar plan. The grey vertices are vertices with doable symbols. The symbols e and b are doable in the interval $[0, 100]$. c and d are exogenous. c is observed at times 40 and 80. d is observed at times 30 and 70. An instance of this plan is valid if it contains an event of type b during $[30, 40]$ or $[70, 80]$, if it contains an event of type e during $[0, 100]$, and if the relative contains are verified. $\{b[32], e[10]\}$ is a valid instance of this Titar plan.

A *Titar plan instance* is a set of events and states which are consistent with a plan i.e. if there is a mapping between the instance's events and the vertices of plan such as all the vertices are valid. A vertex v is said valid at time t if:

- If the symbol of v is exogenous, then there is an event with vertex symbol at time t
- If the vertex symbol is doable, then the vertex symbol is doable at time t, or there is an event with vertex symbol at time t.
- All the states of the set of states states_v are true at time t.
- The absolute constraint of the vertex is true a time t i.e. $\text{Acst}_v(t) = 1$.
- For all edges $e_{v \to v'}$ from v to v':
  If e is positive, there is at least one t' with v' validated at time t' and $\text{Rcst}_e(t' - t) = 1$ If e is negative, there is no t' with v' validated at time t' and $\text{Rcst}_e(t' - t) = 1$.

A *positive Titar plan* is a Titar plan without negative edges. A plan can achieve its objective at any time t such that the tree's trunk is valid at time t. The Fig. 2 shows an example of Titar plan.

D. Meta Titar plans

Different rules can have very different temporal precisions, we can't always combine several temporal rules into a simple list of action to perform, and expect to have a high confidence plan (even if the rules have high confidence). The example III.6 illustrates this problem.

Example III.6. *Suppose two Tita rules:*

$$r_1 := \langle A, \emptyset\rangle \Rightarrow B\,\langle 100\%, \mathbb{U}_{0,100}\rangle$$
$$r_2 := \langle C, \{[B_{-10,-5}, \emptyset]\}\rangle \Rightarrow D\,\langle 100\%, \mathbb{U}_{10,15}\rangle$$

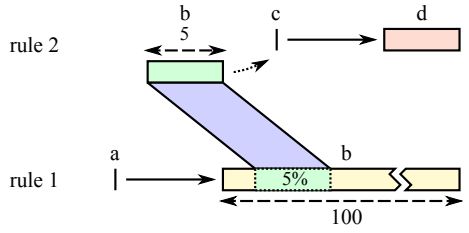

Fig. 3. Illustration of the point discussed in example III.6.

Suppose the objective to be the symbol D. Suppose A and C being the only doable symbols. A plan would be to 'fire' A. (that will cause B) and finally fire C. But r_1 generates B with low (temporal) precision (range of 100), and r_2 need a maximum precision range of 5 to reach a maximum confidence. Therefore, even with two rules having 100% confidence, this plan has a $5\%(=\frac{5}{100})$ confidence to success. The fig. 3 is a graphical representation of this example.

Therefore, we do not combine several Tita rules to make a single Titar plan (which is possible), but we combine several Titar plans to make a Meta Titar plan (i.e. a plan of plans). With the same rules as the previous example, a Meta plan is: Fire A, wait for B to occur, and then fire C. This Meta plan has 100% confidence. Meta Titar plans can express the notion of 'intermediate objectives to wait for' and the disjunction of solution for an objective (or intermediate objectives).

A *Meta Titar plan* is a directed tree graph expressing a set of constraints over Titar plans. Edges are oriented from the head to the leaves. There are two types of vertices: 'solution vertices' and 'problem vertices'. A *solution vertex* is only connected to *problem vertices*. A *problem vertex* is only connected to *solution vertices*. The trunk of the graph is a problem vertex.

Every solution vertex v is labelled with a Titar plan (written $\mathrm{subPlan}_v$). Every problem vertex v' is labelled with a leaf of the plan of v (written $\mathrm{subLeaf}'_v$) such that there is an edge e from v to v', with a Boolean function called the absolute constraint and a symbol. The leaf of a plan connected to a problem vertex of a Meta plan is called a 'caused' vertex.

Example III.7. *Suppose a 'caused' vertex v of a Titar plan p associated to a problem vertex v' of a Meta Titar plan rp. The 'caused' vertex v is a sub-objective of the plan p which is solved by the plans associated to children of v'. The fig 4 represents this example.*

The Fig. 5 shows an example of Meta Titar plan.

We define several types of particulars Meta Titar plans.

A *Positive Meta Titar plans* is a Meta Titar plans that contains only positive Titar plans.

A *Weak linear Meta Titar plans* is a Meta Titar plans such that every plan contains at most one caused vertex. Weak linear Meta Titar plans are a sub class of Meta Titar plans with nice properties for computational issues.

A *Strong linear Meta Titar plans* is a Meta Titar plans with one leaf i.e. it is a path graph. A *Strong linear Meta Titar plans* is also a *weak linear Meta plan*. Every weak linear Meta Titar plan can be decomposed (or unfolded) into a finite set of strong linear Meta Titar plan (see next section for more details).

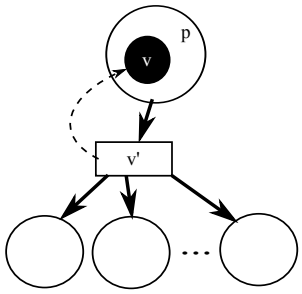

Fig. 4. Representation of the Meta plan rp of example III.7

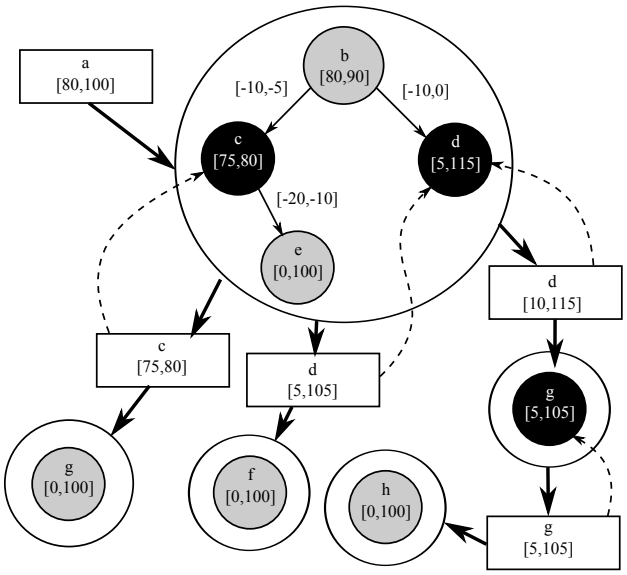

Fig. 5. Example of Meta Titar plan. The main objective is A. The grey vertices are vertices with doable symbols. The black vertices are 'caused' vertices. Dashed arrows represent the labels of the edges. This Meta plan is not (weak or strong) linear.

IV. Planning and Scheduling algorithm

This section present our planning and scheduling algorithm called Titar planner. We suppose that rules (Tita rules) express causal relations.

Similarly to the Strips algorithm [3], our algorithms work with *backward chaining* (or retrograde analysis).

Similarly to the Deviser algorithm [7], our algorithm deals with time windows on actions and goals. The checking of conflicts between positive and negatives part of the plan is time expensive. Therefore, in the case of rules with negations, our algorithm does not check for conflicts between the different parts of the plan. Our solution is to generate instances of plans and check if the goal of the plan is actually generated. This is last part is a heuristic. The algorithm is divided into the four following stages.

V. The four stages

This section presents the four stages of the temporal planning and scheduling algorithm. The fig 6 shows a complete example of Meta planning.

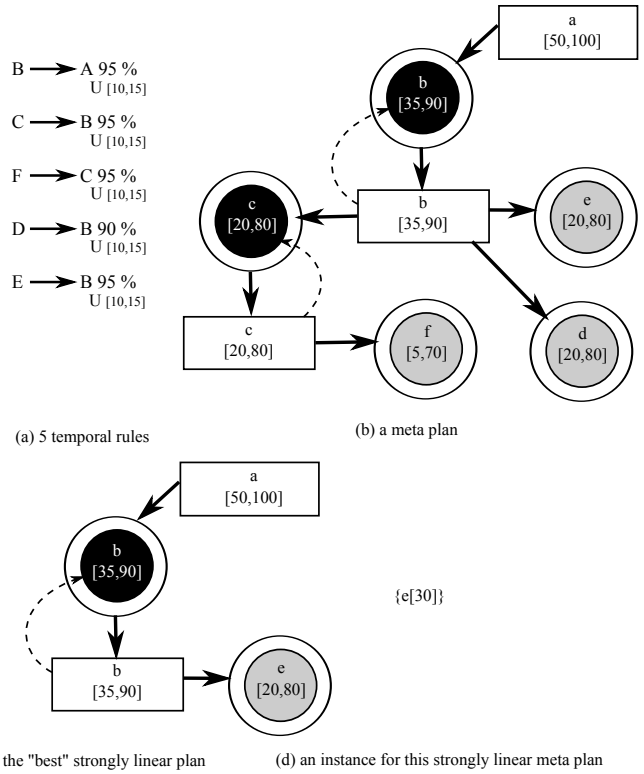

Fig. 6. This figure shows a complete example of Meta planning. The Meta plan (b) is computed from the set of Tita rules (a). Next, a strong linear Meta plan (c) is extracted from the Meta plan (b). In this example, the strong linear Meta plan (c) is the strong linear Meta plan with the higher confidence that can be extracted from (b). Finally, an instance (d) is computed from the strong linear Meta plan (c).

A. Stage 1: Titar planner

The Titar planner takes as input a Tita rule, a set of scheduled events and states, a goal symbol and a time window for this goal symbol. It generates a Titar plan describing how to produce the goal symbol in the time window request with the given rule. The given rule should have as head symbol the requested goal symbol. The user provides a parameter defining the minimum confidence the plan to build. The output plan has the same tree structure as the input rule.

This stage is done with three Depth-first search passes: The first one is the building of the plan's structure according to the rule's structure. The second and third passes propagate and check the temporal constraints of the plan. Only two passes of propagation/validations are necessary because of the tree structure of the plan.

The two equations for the propagation of the constraints are the following ones. For every edge $e := v \to v'$ of the plan p:

$$(\mathrm{Acst}_{v'} *(-\mathrm{Rcst}_e))(x) = 0 \Rightarrow \mathrm{Acst}_v(x) = 0 \qquad (3)$$

$$(\mathrm{Acst}_v * \mathrm{Rcst}_e)(x) = 0 \Rightarrow \mathrm{Acst}_{v'}(x) = 0 \qquad (4)$$

Next, we present three simple examples that illustrate the behavior of the Titar planner. In these three examples, the objective is to find a way to generate an event of type A in the interval $R = [50, 100]$. These examples show the underling idea of the computation of a plan with each of these rules.

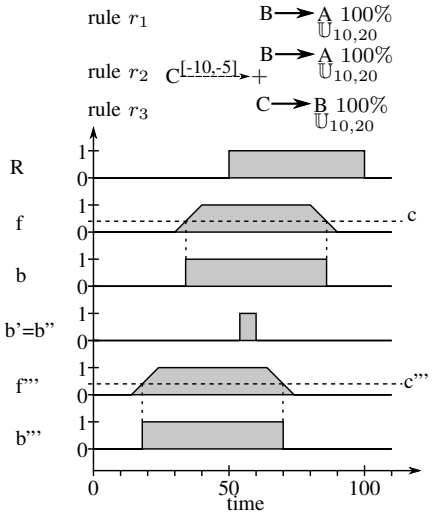

Fig. 7. The rule r_1, r_2, r_3, R, b, f, b', b''', f''' used in the examples V.1, V.2, V.3 and V.4 of behavior of the Titar planner.

Example V.1. *Suppose the rule r_1 introduced in fig. 7. Let $f(t) = ((-\mathrm{dist}_{r_1}) * R)(t)$. If an event of type B occurs at time t, then $f(t)$ is the probability of having an event of type A in the interval R because of the rule r_1. $f(x)$ threshold by $c \in \mathbb{R}$ defines a binary function b. The plan 'trigger an event B at time t with $b(t) = 1$' has a probability of c to generate an event of type A in the interval R. The fig. 7 shows R, b and f.*

Example V.2. *Suppose the rule r_2 introduced in fig. 7. This rule is equivalent to the rule r_1 with an extra condition. Suppose R, f and b as defined in example V.1. In the rule r_2, the temporal constraint between B and C is the Boolean function $m = \mathbb{B}_{-10,-5}$ (i.e. $m(x) = 1$ if and only if $x \in [-10, -5]$). If an event of type B is triggered at time t_1 and an event of type C is triggered at time t_2 with $m(t_2 - t_1) = 1$, then $f(t_1)$ is the probability of having an event of type A in the interval R because of the rule r_2.*

Example V.3. *Suppose R, f, m and b as defined in examples V.1 and V.2. Suppose the rule r_2 introduced in fig. 7. Suppose that events of type C can't be triggered (i.e. C is an exogenous symbol). Suppose that an event of type C occurs at time 50. Let $b' = \mathbb{B}_{55,60}$. In order for the constraint between B and C to be valid, an event B has to occur at time t with $b'(t) = 1$. Let $b''(t) = b(t) \cdot b'(t)$. If an event of type B occurs at time t with $b''(t) = 1$, then $f(t)$ is the probability of having an event of type A in the interval R because of the rule r_2. If an event of type B occurs at time t with $b''(t) \neq 1$, then the rule r_2 can't be applied or the generated event C is not in the R.*

B. Stage 2: Tita Meta planner

The Meta Titar planner takes as input a set of Tita rules, a set of scheduled events and states, a goal symbol and a time window for this goal symbol. It generates a Meta Titar plan that contains the information of how to produce the goal symbol in the time window request with a combination

the given rules. The user provides a parameter defining the minimum confidence the Meta plan to build. The rules to apply do not necessary contain doable symbols. In such case, rules can be used for prediction of exogenous and imprecisely temporally located events (i.e. events such as 'there is 80% chance that an event A will occur between t=10 and t=30' can be rely on as part of a plan).

The stage 2 uses the stage 1 to build Titar plans, and it combines them into a Meta Titar plan. Similarity to the stage 1, this stage is done with three Depth-first search passes: The first one is the building of the Meta plan's structure. The second and third passes propagate and check the temporal constraints of the Meta plan and the plans contained in the Meta plan. Only two passes of propagation/validations are necessary because of the tree structure of the Meta plan. The two conditions for the propagation of the constraints in a Meta plan are the following ones:

1) The absolute constraint of every caused vertex v should be equal to the absolute constraint of the 'problem' vertex v' with is labelled to v i.e. $\mathrm{subLeaf}_{v'} = v$ implies $\mathrm{Acst}_v = \mathrm{Acst}_{v'}$
2) Suppose a 'problem' vertex v. Suppose an edge $e := v \to v'$ with v' a 'solution' vertex labelled with a plan p. Suppose v'' the trunk of the plan p. The two equations of propagation are:

$$((-\operatorname{dist}_r) * \mathrm{Acst}_v)(x) < \frac{minConf}{\mathrm{conf}_r} \Rightarrow \mathrm{Acst}_{v''}(x) = 0 \quad (5)$$

$$(\operatorname{dist}_r * \mathrm{Acst}_{v''})(x) < \frac{minConf}{\mathrm{conf}_r} \Rightarrow \mathrm{Acst}_v(x) = 0 \quad (6)$$

With r the rule of the plan p and $minConf$ the minimum confidence of the plan p.

$$\mathrm{Acst}_{v''} : x \mapsto \begin{cases} 1 & \text{if } ((-\operatorname{dist}_r) * \mathrm{Acst}_v)(x) \geq \frac{minConf}{\mathrm{conf}_r} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{Acst}_v : x \mapsto \begin{cases} 1 & \text{if } (\operatorname{dist}_r * \mathrm{Acst}_{v''})(x) \geq \frac{minConf}{\mathrm{conf}_r} \\ 0 & \text{otherwise} \end{cases}$$

with r the rule of the plan p and $minConf$ the minimum confidence of the plan p

The following example illustrates the behavior the Meta Titar planner. It shows the combination of two rules into a Meta Titar plan.

Example V.4. *Suppose the two rules r_3 and r_4 displayed in fig. 7. Similarly to the last example, the objective is to generate an event of type A in the interval $R = [50, 100]$. Suppose that C is a doable symbol but that B is an exogenous symbol. The functions of this example are displayed in the fig. 7.*

First, the planner tries to apply the rule r_3 to generate A in the interval R. The condition for r_3 to be applied is to get an event of type B at time t with $b(t) = 1$, with b as defined in the example V.1 (This step is similar to the example V.1). Since B is an exogenous symbol, it needs to be generated by a rule.

The same process is applied again with the rule r_4 to generate an event of type B at time t with $b(t) = 1$. Let's define $f'''(t) = ((-\operatorname{dist}_{r_4}) * b)(t)$, and b''' the threshold of the function f''' with $c''' \in \mathbb{R}$. If there is an event C at time t' with $b'''(t') = 1$, then the rule r_4 generates an event of type B at time t with $b(t) = 1$, and then the rule r_3 generates an event of type A in the interval R.

C. Stage 3: Meta plan linearization

The linearization (of a Meta plan) is the extraction of the valid *strong linear Meta plan* (see definition is section III-A) with the highest confidence. Strong linear Meta plans are the intermediate step into the computation of a (Meta) plan instance. This stage requires as input a weak linear Meta plan without negation of doable symbols.

With a Death-first enumeration and a Branch-and-Bound, the algorithm select the path (strong linear Meta plan) of the input Meta plan (a Meta plan has a tree graph structure) with the highest confidence. The confidence of a path is the product of the confidences of the plans associated with its vertices. In the worst case, the number of paths of a tree graph is equal to the number of nodes minus one.

If the algorithm is fed with a weak linear Meta plan with negations, it can produce invalid plans. A simple heuristic to deal with such Meta plans is to enumerate the Meta plans and test its validity.

D. Stage 4: Building of (Meta) plan instance

Given a plan we can compute an instance of a plan. By convention, an instance of a strong linear Meta plan is an instance of the plan associated with the (unique) leaf of the Meta plan.

To compute such instance, the plan structure is explored with a depth first search. At each doable node n of the plan, an event of type symbol_n and time time_t is added to the instance, such that time_t satisfy the absolute constraint of n and the relative constraint of the parent of n. For each node of the plan, a time-sampled has to be select from a set of possible values. Depending on the domain of application, various solutions can be chosen: The smallest value, the median, the mean, the 'more stable', etc.

VI. Experiment

This experiment shows one of the possible uses of the planning algorithm coupled with a learning algorithm. A robot located in an unknown world, will learn the rules of the world, and use this knowledge to achieve its goals. With the Titarl algorithm [1], the robot has the ability to understand the world. With the Titarl Planner algorithm, the robot has the ability to generate plans to reach its goals. There is a large community studying robot cognitive architectures. Since the elaboration of a robot cognitive architecture is not our primary concern in this work, we designed an extremely simple cognitive architecture.

In this experiment, we assume that the world is non deterministic (an event cannot be predicted with 100% confidence), the world is inaccurate (predictions can only be made on

time range and not on time point), and it has unpredictable processes running i.e. if the robot does nothing, some events will still occur.

The world is composed of three buttons (called b_1, b_2 and b_3), three lights (called l_1, l_2 and l_3) and a treat dispenser. The goal of the robot is to get treats. Buttons, lights and the treat dispenser are connected with mechanisms with various different levels of complexity and uncertainty. These mechanisms are unknown from the robot. The hidden "rules" of the world are presented in the next section.

A. Virtual world rules

The rules of the world are the following ones:

- r_0 **:** The lights l_1, l_2 and l_3 are flashing randomly with a respective average of a flash every 50, 40 and 100 seconds.
- r_1 **:** There is 50% chance that a treat is given to the robot between 3 and 4 seconds after the button b_1 is pressed, if the button b_1 was not pressed in the previous 20 seconds.
- r_2 **:** There is 90% chance that the light l_2 flashes between 0 and 10 seconds after the button b_3 is pressed, if the button b_3 was not pressed in the previous 2 seconds.
- r_3 **:** There is 90% chance that a treat is given to the robot between 3 and 4 seconds after the button b_2 is pressed, if the light l_2 has flashed in the previous 3 seconds, and if the button b_2 was not pressed in the previous 3 seconds.

The rule r_0 gives an external noise in the world that the robot cannot predict.

This rule r_1 directly correlates an action to the reward. This rule is the simplest rule that predict the giving of treat. However, this rule has a low probability (50%) and it cannot be used more than once every 20 seconds.

The rule r_2 combined with the rule r_3 can be combined to get a treat. This combination is the best solution to maximize the number of treats. These two rules are not perfectly accurate (r_2 and r_3 have respectively a 10 and 3 seconds range). The combination of these two rules is partially similar to the example III.6: The optimal plan is to trigger the rule r_2, wait for the light l_2 to be activated (l2 might never be activated), and when (and if) l_2 is activated, to trigger the rule r_3. This combination is especially complex because the robot has to wait for the light l_2, but it has no guarantee that it will actually happen. Such Meta plan has optimally a confidence of $81\% = 90\% \times 90\%$. If the robot combines these two rules into a plan (not a Meta plan), then the resulting plan would have a confidence of $8.1\% = 90\% \times 90\% \times \frac{4-3}{10-0}$.

Possibly, the robot can also wait for the rule r_0 to tiger the light l_2, and use the rule r_3 to get a treat.

B. Robot cognitive policy

The robot cognitive policy is composed of two states: the exploration state and the planning state.

Initially, the robot does not know the world and will only do random actions at random times. The action to perform is chosen with a random uniform selection through the possible robot actions. The amount of time the robot waits between actions is also uniform random between 0 and 20 seconds. After a period of random acting, the robot analyses its behavior and the behavior of the world with the Titarl algorithm. It extracts a set of Tita rules and enters the planning state.

In this state, ten times by seconds, the robot builds and executes a linear Meta Titar plan with the objective of getting food in the next 30 seconds, based on the learned rules. The robot only stores the plan instances (the robot does not store/keep in mind the all linear Meta plan). The Meta plan is recomputed every time. This process is a simple architecture with the ability to react to the uncertainty of the world e.g. if, while a plan is executed, a new and better plan appears to be possible (because of the external sources of randomness of the world for example), the old plan will be replaced by the new one. This aspect is discussed in the conclusion of the experiment.

TABLE I

DURATION, NUMBER OF EVENTS, NUMBER OF TRIGGERED RULES AND NUMBER OF TREATS OF THE EXPLORATION AND PLANNING PHASES OF THE EXPERIMENT. THE ABBREVIATION "N.O." REPLACE "NUMBER OF".

Phase	Duration	N.o. events		Rule tigers	Treats
Exploration	10000s	b1: 554 b2: 622 b3: 545	l1: 211 l2: 590 l3: 258	r1: 183 r2: 529 r3: 99	190
Planning	5000s	b1: 0 b2: 335 b3: 365	l1: 98 l2: 375 l3: 117	r1: 0 r2: 365 r3: 335	301

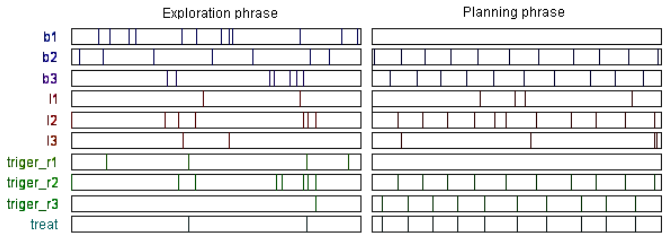

Fig. 8. Extract of the exploration and planning phrase.

C. Results of the experiment

Table I shows the duration, number of events, number of triggered rules and number of treats of the exploration and planning phases of the experiment.

The figure 8 presents an exact of the record of the exploration and planning phases. The events $trigger\_r1$, $trigger\_r2$ and $trigger\_r3$ represent the triggering of the rules r_1, r_2 and r_3. Of course, the robot does not have access to these signals.

The fig. 9 shows the Meta plans generated during the planning phase. The fig. 10 shows the linear Meta plan extracted from this Meta plan.

D. Conclusion

The first observation about the experiment is that the robot successfully understood the hidden rules of the world, and it successfully uses them to get treats.

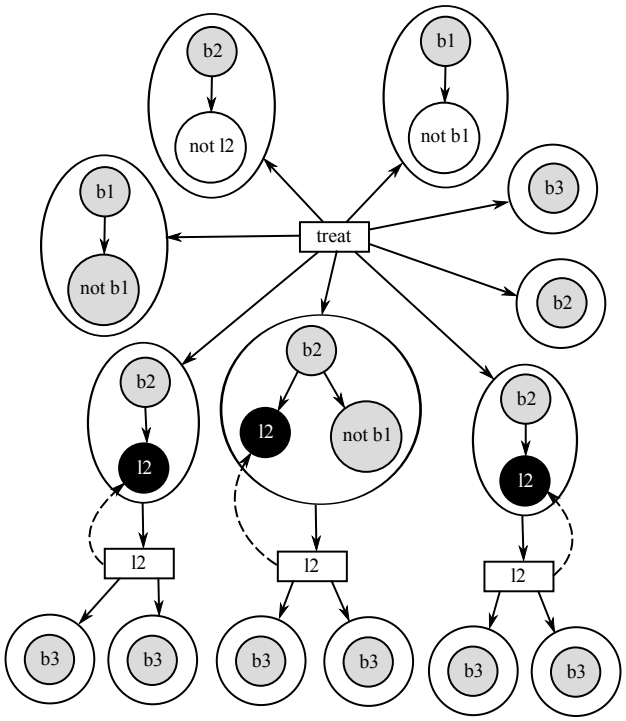

Fig. 9. Extract of the (non linear) Meta plan built. The main interest of this figure is to show the global structure of the graph. The relative and absolute constraints are not displayed.

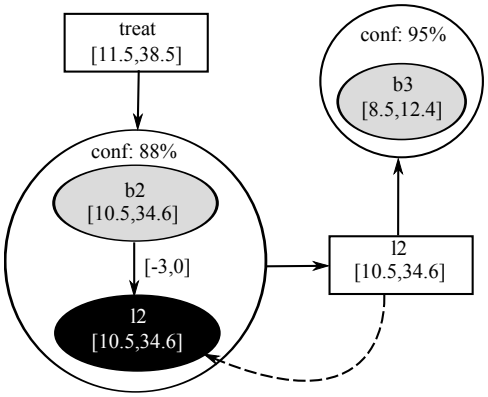

Fig. 10. Representation of the linear Meta plan. The goal is to produces an event 'treat' in the interval [8.5,38.5] when the current time is 8.5 (No actions can't be done before 8.5 i.e. in the past). The global confidence of the Meta plan is the product of the nodes' confidences i.e. $95\% \cdot 88\% = 84\%$.

Second, the robot did not extract the exact rules of the world: The negative parts of the conditions of the rules r_2 and r_3 are not correctly understood. However, the robot reaches its goal with its approximate rules. The analyses of the rules learned by the robot shows that the negation of the rules r_2 and r_3 are actually detected but the time intervals associated with the conditions are not correctly estimated. Because of this incorrect estimation, the rules with negations have low confidence and they are discarded by the planning algorithm.

Other simulations with longer exploration phase or with less activation of the rule r_0 (external random source) shows that the robot can actually correctly learn the negations of the rules r_2 and r_3 with larger or less noisy training examples. This is an example of how an approximate/incorrect representation of a system (the rules are not perfectly learned) can still be good enough to accomplish some goals.

VII. Discussion

This work is based on the specification of a temporal pattern model initially developed for temporal learning. We believe that the tree structure of our pattern model (by opposition to the complete graph structure found in most of the literature [10], [11]) is an interesting trade-off between the power of expression of the pattern, and the complexity to learn it.

From the point of view of automated planning, the tree structure is also easier to solve than the complete graph structure [6]. We present a fast planning algorithm which is able to deal with uncertainty (expressed with probabilities), temporal inaccuracy (expressed with ranges of value and probability distribution), continuous time, and discontinuous (or disjunctive) time constraints (e.g. $[1, 2.2] \cup [3.8, 5]$).

Finally, our experiment of combining a temporal learning algorithm and a planning algorithm into a cognitive architecture is simplistic. However, this is an important first stage that leads us to discover and solve an important number of challenges. Our experiment also demonstrates the feasibility of learning and planning in uncertainty and noisy environments.

References

[1] M. Guillame-Bert and J. L. Crowley, "New approach on temporal data mining for symbolic time sequences: Temporal tree associate rules," in *Proceedings of the 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, ser. ICTAI '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 748–752.

[2] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artif. Intell.*, vol. 90, no. 1-2, pp. 281–300, Feb. 1997.

[3] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[4] H. A. Kautz, B. Selman, and J. Hoffmann, "SatPlan: Planning as satisfiability," in *Abstracts of the 5th International Planning Competition*, 2006.

[5] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artif. Intell.*, 1991.

[6] M. Balaban and T. Rosen, "Stcsp : structured temporal constraint satisfaction problems," *Annals of Mathematics and Artificial Intelligence*, vol. 25, pp. 35–67, January 1999.

[7] S. A. Vere, "Planning in time: Windows and durations for activities and goals," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-5, no. 3, pp. 246 –267, may 1983.

[8] P. J. Schwartz and M. E. Pollack, "Planning with disjunctive temporal constraints," *ICAPS04 Workshop on Integrating Planning into Scheduling*, 2000.

[9] T. Vidal and H. Fargier, "Handling contingency in temporal constraint networks: from consistency to controllabilities," *Journal of Experimental and Theoretical Artificial Intelligence*, 1999.

[10] C. Dousson and T. V. Duong, "Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems," in *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 620–626.

[11] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge discovery*, vol. 1, pp. 259–289, 1997.