Linear Time Logic, Conditioned Models and Planning with Incomplete Knowledge

Marta Cialdea Mayer and Carla Limongelli

Università di Roma Tre, Dipartimento di Informatica e Automazione

Abstract. The "planning as satisfiability" paradigm, which reduces solving a planning problem P to the search of a model of a logical description of P, relies on the assumption that the agent has complete knowledge and control over the world. This work faces the problem of planning in the presence of incomplete information and/or exogenous events, still keeping inside the "planning as satisfiability" paradigm, in the context of linear time logic.

We give a logical characterization of a "conditioned model", which represents a plan solving a given problem together with a set of "conditions" that guarantee its executability. During execution, conditions have to be checked by means of sensing actions. When a condition turns out to be false, a different "conditioned plan" must be considered. A whole conditional plan is represented by a set of conditioned models. The interest of splitting a conditional plan into significant sub-parts is due to the heavy computational complexity of conditional planning.

The paper presents an extension of the standard tableau calculus for linear time logic, allowing one to extract from a single open branch a conditioned model of the initial set of formulae, i.e. a partial description of a model and a set of conditions U guaranteeing its "executability". As can be expected, if U is required to be minimal, the analysis of a single branch is not sufficient. We show how a global view on the whole tableau can be used to prune U from redundant conditions. In any case, if the calculus is to be used with the aim of producing the whole conditional plan off-line, a complete tableau must be built. On the other hand, a single conditioned model can be used when planning and execution (with sensing actions) are intermingled. In that case, the requirement for minimality can reasonably be relaxed.

1 Introduction

The specification of a planning problem P consists of an initial state, a goal to be achieved and a set of actions that the agent can execute. A (linear) plan solving P is a sequence of actions that, when executed, lead from the initial state to a goal state. Most approaches to planning work under the assumption that the agent has complete knowledge of the world, its actions have deterministic effects and the world changes are caused only by the agent's actions. Conditional planning faces problems where such assumptions are dropped. The approach followed by most of the recent work on this area consists in extending partial-order planning algorithms to find "conditional plans" (see, for instance, [9, 11]). Some logical approaches to conditional planning have also been proposed: either based on deduction [4, 8, 13], or on model checking [3, 10].

An effective logical approach to (non-conditional) planning is based on the "planning as satisfiability" paradigm, where planning is reduced to a satisfiability problem, either in classical propositional logic [6, 7] or linear time logic (LTL) [1, 2]: a planning problem can be described by a set Σ of formulae, in such a way that any model of Σ describes a plan solving the problem and vice-versa.

For instance, let us consider the problem where the agent has to enter a room, passing through a door, that is initially closed. Following the methodology proposed in [1, 2], the problem can be described in LTL by means of the following set of formulae:

$$
\begin{aligned}
\Sigma = \{\ & \Box(\mathit{do\_pass} \rightarrow \mathit{open}),\\
& \Box(\mathit{do\_open} \rightarrow \neg \mathit{open}),\\
& \Box(\bigcirc \mathit{in} \equiv (\mathit{in} \wedge \neg \mathit{do\_pass}) \vee (\neg \mathit{in} \wedge \mathit{do\_pass})),\\
& \Box(\bigcirc \mathit{open} \equiv \mathit{open} \vee \mathit{do\_open}),\\
& \neg \mathit{in} \wedge \neg \mathit{open},\\
& \Diamond \mathit{in}\ \}
\end{aligned}
$$

The first two formulae represent the preconditions for the executability of the actions *do_pass* and *do_open*, respectively. The third and fourth formulae are the reformulation in LTL of Reiter's "successor state axioms" [12]. The fifth formula represents the agent's knowledge about the initial state, and the last one represents the goal. Any model of Σ represents a plan solving the problem. For instance, the model where *do_open* is true at state 0, *do_pass* is true at state 1 and *in* is true at state 2. The plan extracted from such a model is the sequence of actions $\langle \mathit{do\_open}, \mathit{do\_pass} \rangle$.

However planning by satisfiability relies on the assumption that the agent has complete knowledge and control over the world. Let us consider, for instance, the same problem as before, but where the agent does not know whether the door is initially open or closed. The problem is represented in LTL by the set Σ_0, obtained from Σ by replacing the initial state formula $\neg \mathit{in} \wedge \neg \mathit{open}$ with $\neg \mathit{in}$. Since there is no information about the state of the door in the initial situation, no model of Σ_0 represents a correct plan.

Incomplete knowledge about the initial state is not the only source of uncertainty. When the world is not under the agent's complete control, the state of the world at a given time point cannot be predicted simply on the base of what holds before and the agent's actions. For example, let us assume that the agent can open the door but the effect of the action is guaranteed for the next state only, for instance because other agents may pass by and open or close the door; the agent must enter the room and exit again. The problem is represented by the set Σ_1 obtained from Σ_0 by relaxing the successor state axiom for *open* and modifying the goal formula:

$$\begin{aligned}
\Sigma_1 = \{ \; & \Box(do_pass \rightarrow open), \\
& \Box(do_open \rightarrow \neg open), \\
& \Box(\bigcirc in \equiv (in \wedge \neg do_pass) \vee (\neg in \wedge do_pass)), \\
& \Box(do_open \rightarrow \bigcirc open), \\
& \neg in, \\
& \Diamond(in \wedge \Diamond \neg in) \; \}
\end{aligned}$$

In such cases, conditional planning solves the problems by means of *conditional plans*. For instance, a conditional plan solving the problem represented by Σ_1 is:

IF the door is open
THEN pass; afterwards, IF the door is open
                                        THEN pass
                                        ELSE open the door and pass
ELSE open the door and pass;
          afterwards, IF the door is open
                              THEN pass
                              ELSE open the door and pass

The execution of a conditional plan requires sensing: during execution, at each IF/THEN/ELSE branching, the agent checks the state of the world and follows the respective branch. A conditional plan can be represented by a tree: each node N represents a property that must be checked at execution time, the left subtree of N is the conditional plan that must be executed if N turns out to be true, the right subtree is the plan to be executed when N is false.

A conditional plan P can be seen as a set of "*conditioned plans*", each corresponding to a branch in the tree representing P. For instance, the conditional plan above can be split into the following set of "complementary" conditioned plans:

1) IF the door is open THEN pass;
   afterwards, IF the door is open THEN pass
2) IF the door is open THEN pass;
   afterwards, IF the door is closed THEN open the door and pass
3) IF the door is closed THEN open the door and pass;
   afterwards, IF the door is open THEN pass
4) IF the door is closed THEN open the door and pass;
   afterwards, IF the door is closed THEN open the door and pass.

Each conditioned plan can in turn be described by means of a linear plan (a sequence of actions) together with a set of conditions guaranteeing the executability of the plan. For instance, the third plan above can be represented by the pair:

$\langle\ \langle$ do_open, do_pass, do_pass $\rangle$,
  { the door is closed at state 0, the door is open at state 2 } $\rangle$

The interest of splitting a conditional plan into significant sub-parts is due to the heavy computational complexity of conditional planning [13]. Instead of producing the whole conditional plan off-line, and then executing it, we can consider the possibility that planning and execution (with sensing actions) are intermingled. In that case, a single conditioned plan can be generated at first. During its execution, its conditions are checked by means of sensing actions. When a condition turns out to be false, a different (complementary) conditioned plan is generated and considered. Conditional planning can be reduced to the search for a set of complementary conditioned plans.

In this work we give a logical characterization of a "conditioned model" of a set Σ of LTL-formulae (describing a planning problem) (Section 2). A conditioned model, which represents a conditioned plan, is a pair $\langle S, U\rangle$, where S is a partial description of a temporal model of Σ and U a set of conditions that guarantee the "executability" of S.

In Section 3 we describe a tableau calculus that allows us to extract a conditioned model $\langle S, U\rangle$ of Σ from a single open branch of a tableau for Σ. As can be expected, if U is required to be minimal, the analysis of a single branch is not sufficient. We show (Section 4) how a global view on the whole tableau can be used to prune U from redundant conditions. In any case, if the calculus is to be used with the aim of producing the whole conditional plan off-line, a complete tableau must be built. Section 5 concludes this work.

2 Conditioned Models

We consider the language of linear time logic over a set of propositional letters $\mathbb{P}$ built by means of the connectives $\neg$, $\wedge$, $\vee$ and the future time $\Box$ (always), $\Diamond$ (eventually), $\bigcirc$ (next). Implication and double implication are defined symbols. For convenience, we restrict the language to formulae in negation normal form, and consider negation over non atomic formulae as a defined symbol too:

$$\neg(A \vee B) \equiv_{def} \neg A \wedge \neg B \qquad \neg(A \wedge B) \equiv_{def} \neg A \vee \neg B \qquad \neg\neg A \equiv_{def} A$$
$$\neg\Box A \equiv_{def} \Diamond\neg A \qquad \neg\Diamond A \equiv_{def} \Box\neg A \qquad \neg\bigcirc A \equiv_{def} \bigcirc\neg A$$

For our aims, a non standard way of describing the semantics of LTL is more convenient. It makes use of *labelled formulae*, i.e. expressions of the form $n : A$ where $n \in \mathbb{N}$ and A is an LTL-formula. If ℓ is a literal, then $n : \ell$ is a labelled literal. A set S of labelled literals is consistent if it does not contain both $n : p$ and $n : \neg p$ for any $n \in \mathbb{N}$ and $p \in \mathbb{P}$. It is complete if for all $n \in \mathbb{N}$ and all $p \in \mathbb{P}$, either $n : p \in S$ or $n : \neg p \in S$.

A temporal interpretation $\mathcal{M}$ is a consistent and complete set of labelled literals. If $\mathcal{M}$ is an interpretation of $\mathbb{P}$, $n \in \mathbb{N}$ and A is a formula, the relation $\mathcal{M} \models n : A$ corresponds to the usual satisfiability relation $\mathcal{M}_n \models A$ (the n-th state of $\mathcal{M}$ satisfies A) and is recursively defined as follows:

1. $\mathcal{M} \models n : \ell$ iff $n : \ell \in \mathcal{M}$, when ℓ is a literal;
2. $\mathcal{M} \models n : A \wedge B$ iff $\mathcal{M} \models n : A$ and $\mathcal{M} \models n : B$;

3. $\mathcal{M} \models n : A \vee B$ iff either $\mathcal{M} \models n : A$ or $\mathcal{M} \models n : B$;
4. $\mathcal{M} \models n : \bigcirc A$ iff $\mathcal{M} \models n+1 : A$;
5. $\mathcal{M} \models n : \Box A$ iff for all $k \geq n$, $\mathcal{M} \models k : A$;
6. $\mathcal{M} \models n : \Diamond A$ iff for some $k \geq n$, $\mathcal{M} \models k : A$.

An LTL-formula A is true in $\mathcal{M}$ (and $\mathcal{M}$ is a model of A) iff $\mathcal{M} \models 0 : A$. If Σ is a set of formulae, $\mathcal{M} \models \Sigma$ iff $\mathcal{M} \models A$ for all $A \in \Sigma$. From now on, a set Σ of LTL-formulae is considered equivalent to $\{0 : A \mid A \in \Sigma\}$.

The rest of this section introduces the notion of conditioned model and its main desirable properties.

Definition 1. *If S is a set of labelled formulae, the* history *of S up to n, $h(S, n)$, is*

$$\{k : \ell \mid k : \ell \in S,\ \ell \text{ is a literal, and } k \leq n\}$$

Let S_0 be a set of formulae representing a planning problem, according to the method proposed in [2] (but for the fact that incomplete information is allowed). In particular, we assume that the agent's tasks and goals are represented by formulae of the form $\Diamond G$, where G is classical (the approach can easily be extended to nested goals, assuming only that G does not contain any occurrence of the $\Box$ operator). If $\mathcal{M}$ is a model of S_0, we are interested in extracting a set of conditions ensuring the executability of the plan represented by $\mathcal{M}$, to solve the problem represented by S_0. In other words, we want to "explain" the executability of $\mathcal{M}$. In general, however, it is not the whole model $\mathcal{M}$ that has to be explained, but only a finite fragment of the model, up to the achievement of the goals. The following definitions formalize this concept.

Definition 2. *Let S_0 be a set of labelled formulae, $\mathcal{M}$ a model of S_0, $S \subseteq \mathcal{M}$ and $U \subseteq S$. Then U is a (sufficient)* set of conditions *explaining S in the context of S_0 (briefly, U explains S in S_0) iff*

for all $n : \ell \in S - U$: $S_0, h(U, n-1) \models n : \ell$.

The condition above amounts to saying that, for all $n : \ell \in S - U$ and for every model $\mathcal{M}'$ of S_0, if $\mathcal{M}'$ is also a model of the subset $h(U, n-1)$ of the history of U up to $n-1$, then $\mathcal{M}' \models n : \ell$, too. In informal terms, $h(U, n)$ "explains", in the context of S_0, all that happens, according to S, up to the n-th state. If S implies that the goals are achieved before the n-th step, then $h(U, n-1)$ actually guarantees the achievement of the goals.

Definition 3. *Let S_0 be a set of labelled formulae representing a planning problem. A* conditioned model *of S_0 is a pair $\langle S, U\rangle$, where $S \subseteq \mathcal{M}$ for some model $\mathcal{M}$ of S_0, U explains S in S_0 and S satisfies the goals in S_0, i.e. for all $0 : \Diamond G \in S_0$, $S \models n : G$ for some n.*

A conditioned plan solving the problem encoded by Σ can be extracted from the set U of conditions of a conditioned model of Σ, by dropping negative literals representing actions and splitting the rest of U into the set of atoms representing actions and the fluents (conditions guaranteeing the executability of the plan).

For instance, let us consider the set Σ_1 of Section 1, and the following subset S of one of its models:

$$\begin{aligned} S = \{\; & 0:\mathit{open},\ 0:\mathit{do\_pass},\ 0:\neg\mathit{do\_open},\ 0:\neg\mathit{in}, \\ & 1:\neg\mathit{open},\ 1:\neg\mathit{do\_pass},\ 1:\mathit{do\_open},\ 1:\mathit{in}, \\ & 2:\mathit{open},\ 2:\mathit{do\_pass},\ 2:\neg\mathit{do\_open},\ 2:\mathit{in}, \\ & 3:\mathit{open},\ 3:\neg\mathit{do\_pass},\ 3:\neg\mathit{do\_open},\ 3:\neg\mathit{in}\} \end{aligned}$$

S implies that all the goals are achieved at the third state. The set:

$$\begin{aligned} U = \{\; & 0:\mathit{open},\ 0:\mathit{do\_pass},\ 0:\neg\mathit{do\_open} \\ & 1:\neg\mathit{open},\ 1:\neg\mathit{do\_pass},\ 1:\mathit{do\_open}, \\ & 2:\mathit{do\_pass},\ 2:\neg\mathit{do\_open}, \\ & 3:\mathit{open},\ 3:\neg\mathit{do\_pass},\ 3:\neg\mathit{do\_open}\} \end{aligned}$$

is a set explaining S in Σ_1. In fact:

$$\begin{aligned} & S - U = \{0:\neg\mathit{in},\ 1:\mathit{in},\ 2:\mathit{open},\ 2:\mathit{in},\ 3:\neg\mathit{in}\} \\ & h(U,-1) = \emptyset \\ & h(U,0) = \{0:\mathit{open},\ 0:\mathit{do\_pass},\ 0:\neg\mathit{do\_open}\} \\ & h(U,1) = h(U,0) \cup \{1:\neg\mathit{open},\ 1:\neg\mathit{do\_pass},\ 1:\mathit{do\_open}\} \\ & h(U,2) = h(U,1) \cup \{\mathit{do\_pass},\ 2:\neg\mathit{do\_open}\} \end{aligned}$$

and

$$\begin{array}{ll} \Sigma_1 \models 0:\neg\mathit{in} & \Sigma_1,\ h(U,1)\ \models 2:\mathit{in} \\ \Sigma_1,\ h(U,0)\ \models 1:\mathit{in} & \Sigma_1,\ h(U,2)\ \models 3:\neg\mathit{in} \\ \Sigma_1,\ h(U,1)\ \models 2:\mathit{open} & \end{array}$$

Omitting the negative "action" literals, U can be read as saying:

IF $0:\mathit{open}$, $0:\mathit{do\_pass}$, $1:\neg\mathit{open}$, $1:\mathit{do\_open}$, and $2:\mathit{do\_pass}$, THEN the goal is achieved.

The conditioned plan that is extracted from U is then:

$\langle 0:\mathit{do\_pass},\ 1:\mathit{do\_open},\ 2:\mathit{do\_pass}\rangle$ if $\{0:\mathit{open},\ 1:\neg\mathit{open}\}$

Obviously, if $U = S$, then $\langle S, U\rangle$ is trivially a conditioned model of Σ. So, we are interested in finding "non trivial" sets of conditions. This issue is addressed in Section 4.

3 Conditional Tableaux

This section introduces a tableau system that, given a set Σ of satisfiable LTL-formulae, allows one to compute a conditioned model of Σ. The calculus is mainly a labelled version of the standard tableau system for LTL [16]. Moreover, to each labelled formula is associated a set of "conditions" C, in such a way that, when a non empty set C is associated to $n:\ell$, then $n:\ell$ is included in the set of conditions to be computed.

Tableau nodes are *conditioned labelled formulae* (cl-formulae), i.e. expressions of the form $n : A$ *if* C, where $n : A$ is a labelled formula and C is a set of labelled formulae (parentheses will sometimes enclose cl-formulae to enhance readability). A cl-formula $n : A$ *if* $\emptyset$ is abbreviated by $n : A$. If Σ is a set of LTL-formulae, the initial tableau for Σ is the one-branch tableau whose nodes are $0 : A$ for all $A \in \Sigma$.

The next definition introduces some useful notions.

Definition 4.

1. *If K is a tableau branch, then*

$$forms(K) = \{n : A \mid (n : A \text{ if } C) \in K \textit{ for some } C\}$$

2. *The notion of history is generalized to sets K of cl-formulae as follows:*

$$h(K, n) = \{k : \ell \mid k : \ell \in forms(K),\ \ell \textit{ is a literal, and } k \leq n\}$$

Tableaux are expanded by means of the rules displayed in Figure 1. In the $\bigcirc$-rule, K is the branch being expanded. Note that the call to a theorem prover for LTL required by such a rule can be expected to face a quite simple problem, since $h(K, n)$ is a set of cl-literals.

Next we define when a branch is complete and when it is closed, what is the set $U(K)$ extracted from a branch, and the temporal interpretation represented by a branch.

Definition 5. *Let K be a tableau branch.*

1. *K is completely expanded at n (or n-*complete*) iff every non-literal formula labelled with $k \leq n$ in K has been expanded.*
2. *K is* looping *if there are m, p with $p > 0$ such that K is completely expanded at $m + p$ and*

$$\{A \mid m : \bigcirc A \in forms(K)\} = \{A \mid m + p : \bigcirc A \in forms(K)\}$$

Then K is said to be "periodic" with starting index $m + 1$ and period p.

3. *K is complete if it is either looping or n-complete for all n.*
4. *K contains an* unsatisfied eventuality *if $n : \Diamond A \in forms(K)$ and:*
   - *(a) either K is p-complete for all p or periodic with starting index $m \geq n$, and $k : A \notin forms(K)$ for all $k \geq n$;*
   - *(b) or K is periodic with starting index $m < n$ and period p, and $k : A \notin forms(K)$ for all k such that $m \leq k \leq m + p$.*
5. *K is* closed *if one of the following conditions holds:*
   - *(a) $forms(K)$ contains a formula $n : A$ and its negation $n : \neg A$.*
   - *(b) K is complete and it contains some unsatisfied eventuality.*
6. *$lits(K) = \{n : \ell \mid \ell$ is a literal and $n : \ell \in forms(K)\}$*
7. *$U(K) = \{n : \ell \mid \ell$ is a literal, $n : \ell \notin K$ and $(n : \ell$ if $C) \in K$ for some $C \neq \emptyset\}$*

$$\frac{n : A \wedge B \ \textit{if}\ C}{\begin{array}{c} n : A \ \textit{if}\ C \\ n : B \ \textit{if}\ C \end{array}} \ (\alpha)$$

$$\frac{n : A \vee B \ \textit{if}\ C}{n : A \ \textit{if}\ C \cup \{n : \neg B\} \qquad n : B \ \textit{if}\ C \cup \{n : \neg A\}} \ (\beta)$$

$$\frac{n : \Box A \ \textit{if}\ C}{\begin{array}{c} n : A \ \textit{if}\ C \\ n : \bigcirc\Box A \ \textit{if}\ C \end{array}} \ (\nu)$$

$$\frac{n : \Diamond A \ \textit{if}\ C}{n : A \ \textit{if}\ \{n : A\} \cup C \qquad n : \bigcirc\Diamond A \ \textit{if}\ \{n : \neg A\} \cup C} \ (\pi)$$

$$\begin{array}{c} \vdots\ (K) \\ \dfrac{n : \bigcirc A \ \textit{if}\ C}{n+1 : A \ \textit{if}\ C'} \end{array} \ (\bigcirc) \quad \begin{array}{l} \text{where } C' = \emptyset \text{ if } h(K,n) \models C, \\ \text{otherwise } C' = C. \end{array}$$

Fig. 1. The conditional tableaux expansion rules

8. *$unfold(K)$ is the model (rather, the set of models) represented by K:*

$$unfold(K) = \begin{cases} lits(K) & \textit{if } K \textit{ is not a looping branch} \\ lits(K) \cup \{k+p+i : \ell \mid k+i : \ell \in lits(K), i \in \mathbb{N}\} & \\ & \textit{if } K \textit{ is periodic with starting index } k \\ & \textit{and period } p \end{cases}$$

9. *A tableau is complete iff all its branches are complete. It is closed iff all its branches are closed.*

If K is a complete and open branch in a tableau for K_0, then $unfold(K)$ represents a model of K_0 (see Theorem 1 below) and $U(K)$ is the set of conditions explaining $lits(K)$ in the context of K_0 (Theorem 2). As an example, consider the following set of sentences, representing a simplified planning problem:

$$\Sigma = \{\ \neg in,\ do\_pass \to open,\ \bigcirc in \equiv do\_pass,\ \Diamond in\ \}$$

A branch in a tableau for such a set of sentences shown in Figure 2, where the order in which the nodes are expanded is displayed on the right. The cl-literals in the branch are

$$\{0 : \neg in,\ 0 : open\ \mathit{if}\ \{0 : do\_pass\},\ 0 : do\_pass\ \mathit{if}\ \{0 : \bigcirc in\},\\ 1 : in,\ 1 : in\ \mathit{if}\ \{1 : in\}\}$$

and $U(K) = \{0 : open,\ 0 : do\_pass\}$.

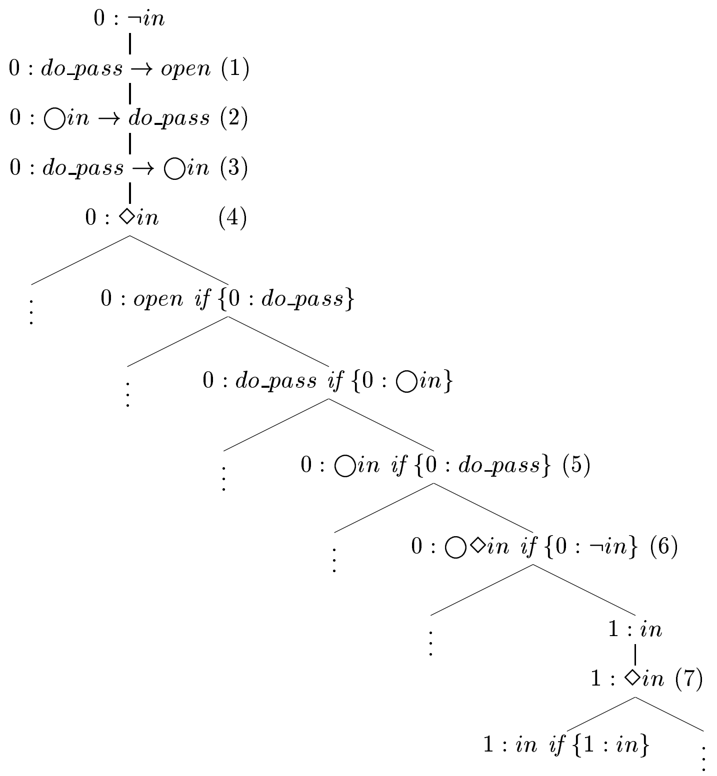

Fig. 2. A branch in a tableau for $\{\neg in, do\_pass \rightarrow open, \bigcirc in \equiv do\_pass, \Diamond in\}$

3.1 Properties of the calculus

The semantics of cl-formulae just ignores the "conditional" part:

$$\mathcal{M} \models (n : A\ \mathit{if}\ C) \quad \text{iff} \quad \mathcal{M} \models n : A$$

The other basic semantical notions are extended to cl-formulae in the obvious way.

It can easily be recognized that, if the C-parts of the tableau nodes are ignored, the calculus is a labelled version of the standard tableau calculus for LTL [16]: the time point at which a given formula is assumed to hold is identified by its numerical label, instead of its "context", i.e. the whole set of formulae labelling a tableau node. From this equivalence it follows that:

Theorem 1. *If $\mathcal{M} \models \Sigma$ then there is a branch K in any tableau for Σ such that $\mathcal{M} \models K$, hence in particular $lits(K) \subseteq \mathcal{M}$. Vice-versa, every complete and open branch K in a tableau for Σ represents a model of Σ: i.e. for all $\mathcal{M}$, if $unfold(K) \subseteq \mathcal{M}$, then $\mathcal{M} \models \Sigma$.*

As a consequence, the calculus is sound and complete in the standard sense. The main concern of this work consists however in showing that $U(K)$ is a set of conditions explaining $lits(K)$ in the context of the initial tableau. The next theorem states in fact that $U(K)$ is large enough.

Theorem 2. *Let K be a complete and open branch in a tableau for K_0. Then $\langle lits(K), U(K)\rangle$ is a conditioned model of K_0.*

Proof. By Theorem 1, $unfold(K) \models K_0$, therefore $lits(K)$ is a subset of a model of K_0. The fact that, for all labelled literal $n : \ell \in K - U(K)$:

$$K_0, h(U(K), n-1) \models n : \ell$$

is proved by showing that, if $n : A$ *if* C is a node in a branch K of a tableau initialized with K_0, then

$$K_0, h(U(K), n-1), C \models n : A$$

The latter assertion can easily be proved by induction on tableaux.

Finally, let $0 : \Diamond G$ be a goal in K_0. Since K does not contain any unsatisfied eventuality, $n : G \in forms(K)$ for some $n \geq 0$, and, since K is complete and G is classical, $lits(K) \models n : G$. Therefore $lits(K)$ satisfies all the goals in K_0.

Assuming that the $\bigcirc$-rule is never applied to a formula $n : \bigcirc A$ before another rule is applied to any formula labelled by n, it can be shown that if $(n : \ell$ *if* $C) \in K$, where K is a tableau branch, and $h(U(K), n-1) \models C$, then $n : \ell \notin U(K)$. However, if K is any complete and open branch in a tableau initialized with K_0, then $U(K)$ is not necessarily a minimal set of conditions explaining $lits(K)$ in the context of K_0, i.e. a minimal subset X of $lits(K)$ such that for all $n : \ell \in lits(K) - X$, $K_0, h(X, n-1) \models n : \ell$. It may even happen that some $n : \ell$ belongs to $U(K)$, for some complete and open branch K in a tableau initialized with K_0, even if $K_0 \models n : \ell$. Consider for instance $K_0 = \{0 : A \vee B,\ 0 : A \rightarrow p,\ 0 : B \rightarrow p\}$. Clearly, $K_0 \models 0 : p$, although $0 : p$ occurs "conditioned" in any branch in a tableau for K_0. Figure 3 shows a branch in a tableau for K_0.

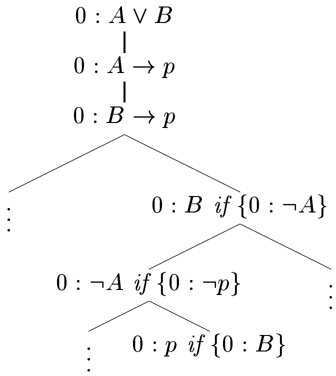

Fig. 3. A branch in a conditional tableau for $\{A \vee B,\ A \to p,\ B \to p\}$

4 Minimality

Let S_0 be a set of labelled formulae, and $\langle S, U\rangle$ a conditioned model of S_0. Then U is a minimal set of conditions explaining S in the context of S_0 iff no proper subset of U explains S in S_0.

Consider for instance the set Σ of Section 3 (page 8) and the corresponding set of labelled formulae:

$$S_0 = \{\ 0 : \neg \mathit{in},\ 0 : \mathit{do\_pass} \to \mathit{open},\ 0 : \bigcirc \mathit{in} \equiv \mathit{do\_pass},\ 0 : \Diamond \mathit{in}\ \}$$

Let S be the following subset of a model of S_0:

$$S = \{0 : \neg \mathit{in},\ 0 : \mathit{open},\ 0 : \mathit{do\_pass},\ 1 : \mathit{in}\}$$

A minimal set X such that for all $n : \ell \in S - X$: $S_0, h(X, n-1) \models n : \ell$ is $X = \{0 : \mathit{do\_pass},\ 0 : \mathit{open}\}$ (X represents the conditioned plan "IF $0 : \mathit{open}$ THEN $0 : \mathit{do\_pass}$"). In fact $S - X = \{0 : \neg \mathit{in},\ 1 : \mathit{in}\}$ and

$$\begin{array}{l} S_0 \models 0 : \neg \mathit{in} \\ S_0,\ 0 : \mathit{do\_pass},\ 0 : \mathit{open} \models 1 : \mathit{in} \end{array}$$

And no subset of X explains S in S_0. In fact, $S_0 \not\models 0 : \mathit{open}$ and $S_0 \not\models 0 : \mathit{do\_pass}$, so that both $0 : \mathit{open}$ and $0 : \mathit{do\_pass}$ must be in X.

Note that the notion of "minimal explanation" adopted here is different from the corresponding notion used in abductive reasoning. In that context, X would not be considered "minimal", since $S_0,\ 0 : \mathit{do\_pass} \models 0 : \mathit{open}$.

The following result shows that minimality of a set U of conditions is equivalent to a property that can be checked "locally", i.e. without considering all the proper subsets of U.

Lemma 1. *If $\langle S, U\rangle$ is a conditioned model of S_0, then U is minimal if and only if for all $n : \ell \in U$: $S_0, h(U, n-1) \not\models n : \ell$.*

Proof. In fact, let us assume that U is not minimal, i.e. for some $n : \ell \in U$ $X = U - \{n : \ell\}$ explains S in S_0. Since $n : \ell \notin X$, $S_0, h(X, n-1) \models n : \ell$. Therefore also $S_0, h(U, n-1) \models n : \ell$ (since $X \subset U$).

Vice versa: let us assume that for some $n : \ell \in U$, $S_0, h(U, n-1) \models n : \ell$. Consider $X = U - \{n : \ell\}$ and any $k : \ell' \in S - X$. If $k = n$ and $\ell' = \ell$, then $S_0, h(X, k-1) \models k : \ell'$ by hypothesis. Otherwise, $k : \ell' \in S - U$, therefore $S_0, h(U, k-1) \models k : \ell'$. Since $S_0, h(U, k-1) \models n : \ell$, then also $S_0, h(U, k-1) - \{n : \ell\} \models k : \ell'$, i.e. $S_0, h(X, k-1) \models k : \ell'$. Therefore U is not minimal.

By Theorem 1, minimality of $U(K)$ can be obtained if the following "simplification rule" is added to the calculus:

$$\frac{\begin{array}{c}\vdots\ (K)\\ n : \ell \ \textit{if}\ C\end{array}}{n : \ell} \qquad \begin{array}{l}\text{if } K_0, h(K, n-1) \models n : \ell,\\ \text{where } K_0 \text{ is the initial tableau}\end{array}$$

However, the recursive call to the theorem prover required by this rule would be quite expensive.

Here we present an alternative method, consisting in filtering $U(K)$ with information obtained by a global view on the tableau. Intuitively, in order to check whether $K_0, h(K, n-1) \models n : \ell$, the tableau for K_0 that is being built is exploited.

The next definition introduces a subset of $U(K)$ which can be proved to be minimal.

Definition 6. *Let $\mathcal{T}$ be a tableau for K_0 and K a complete and open branch in $\mathcal{T}$. Then*

$$U^*(K) = U(K) - \{n : \ell \mid \textit{ for all branches } K_i \textit{ in } \mathcal{T},\ K_i, h(K, n-1) \models n : \ell\}$$

Remark 1. If $n : \ell \in U(K) - U^*(K)$, then $K_0, h(K, n-1) \models n : \ell$. In fact, let us assume (by contraposition) that $K_0, h(K, n-1) \not\models n : \ell$, for some $n : \ell \in U(K)$, and $\mathcal{M}$ is a model of $K_0 \cup h(K, n-1) \cup \{n : \neg\ell\}$. By Theorem 1, there is a branch K_i in $\mathcal{T}$ such that $\mathcal{M} \models K_i$, and consequently $K_i, h(K, n-1) \not\models n : \ell$, i.e. $n : \ell \in U^*(K)$.

We note, first of all, that, in the context of K_0, $h(K, n)$ and $h(U^*(K), n)$ are equivalent:

Lemma 2. *If K is a complete and open branch in a tableau for K_0, then for all $n : \ell \in lits(K)$, $K_0, h(U^*(K), n) \models n : \ell$.*

Proof. By Theorem 2, $lits(K)$ and $U(K)$ are equivalent in the context of K_0, i.e. for all $n : \ell \in lits(K)$, $K_0, h(U(K), n) \models n : \ell$. Therefore, it suffices to show that, in the context of K_0, $U(K)$ and $U^*(K)$ are equivalent, i.e. that for all $n : \ell \in U(K)$, $K_0, h(U^*(K), n) \models n : \ell$. If $n : \ell \in U^*(K)$, trivially

$K_0, h(U^*(K), n) \models n : \ell$. Let us assume then that $n : \ell \in U(K)$ and $n : \ell \notin U^*(K)$, i.e. for all branches K_i in $\mathcal{T}$, $K_i, h(K, n-1) \models n : \ell$. We prove that $K_0, h(U^*(K), n) \models n : \ell$ by induction on n.

1. If $n = 0$ and $K_i \models 0 : \ell$ for all branches K_i in $\mathcal{T}$, then $K_0 \models 0 : \ell$ by Remark 1, since $h(K, n-1) = \emptyset$.
2. If $n > 0$, let us assume (inductive hypothesis) that for all $k < n$ and $k : \ell \in U(K)$, $K_0, h(U^*(K), k) \models k : \ell$, i.e. if $k < n$ then $U(K)$ and $U^*(K)$ are equivalent in the context of K_0. Since, by hypothesis, for all branches K_i in $\mathcal{T}$, $K_i, h(K, n-1) \models n : \ell$, by Remark 1 $K_0, h(K, n-1) \models n : \ell$. By Theorem 2, $K_0, h(U(K), n-1) \models n : \ell$; by the inductive hypothesis, $K_0, h(U^*(K), n-1) \models n : \ell$, since $n-1 < n$, and then also $K_0, h(U^*(K), n) \models n : \ell$.

The next theorem establishes that $U^*(K)$ indeed explains $lits(K)$ in K_0, i.e. that $U^*(K)$ is still "large enough".

Theorem 3. *Let $\mathcal{T}$ be a complete tableau for K_0 and K an open branch in $\mathcal{T}$. Then for all $n : \ell \in K - U^*(K)$, $K_0, h(U^*(K), n-1) \models n : \ell$.*

Proof. Exploiting Lemma 2, it is sufficient to prove that for all $n : \ell \in K - U^*(K)$: $K_0, h(K, n-1) \models n : \ell$. If $n : \ell \in K - U(K)$, this follows from Theorem 2. If $n : \ell \in U(K) - U^*(K)$, then $K_0, h(K, n-1) \models n : \ell$ follows from Remark 1.

Finally, we prove that $U^*(K)$ is minimal:

Theorem 4. *Let $\mathcal{T}$ be a tableau for K_0 and K a complete and open branch in $\mathcal{T}$. Then $U^*(K)$ is a minimal set of conditions for $lits(K)$ in K_0, i.e. for all $n : \ell \in U^*(K)$, $K_0, h(U^*(K), n-1) \not\models n : \ell$.*

Proof. We show that if $K_0, h(U^*(K), n-1) \models n : \ell$, then $n : \ell \notin U^*(K)$. Let us assume that $K_0, h(U^*(K), n-1) \models n : \ell$, hence also $(\star)$ $K_0, h(K, n-1) \models n : \ell$. If $n : \ell \notin U(K)$, then also $n : \ell \notin U^*(K)$. So, it is sufficient to show that if $n : \ell \in U(K)$ then for any branch K_i in the tableau, $K_i, h(K, n-1) \models n : \ell$. Let us assume, by absurd, that for some branch K_i and some interpretation $\mathcal{M}$, $\mathcal{M} \models K_i \cup h(K, n-1) \cup \{n : \neg\ell\}$. Since trivially a model of any branch of the tableau is also a model of a sub-branch, $\mathcal{M} \models K_0 \cup h(K, n-1) \cup \{n : \neg\ell\}$, contradicting $(\star)$.

Computing a minimal set of conditions explaining $lits(K)$ in K_0 is necessarily more expensive than computing a non-necessarily minimal one. A way how this can be achieved is the following: after having built a complete and open branch K and computed $U(K)$, all the other open branches $K_1, ..., K_m$ are considered. For each n such that a literal $n : \ell$ belongs to $U(K)$, $h(K, n-1)$ is added to K_i, for all $i = 1, ..., m$, and the branches obtained in that way are (possibly) expanded further to obtain complete ones. If $n : \ell$ occurs in all the open branches obtained in this way, then $n : \ell$ is eliminated from $U(K)$. In fact, obviously, $K_i, h(K, n-1) \models n : \ell$ if and only iff $n : \ell$ occurs in all the open branches in a tableau for $K_i \cup h(K, n-1)$. This method can be applied in an incremental fashion, considering in turn all the literals labelled by 0, then the literals labelled by 1, and so on. In such a way, the recursive calls to the theorem prover exploit the work already done sofar.

5 Concluding remarks

In this work we have defined the notion of conditioned model of a set of formulae Σ, that consists of a partial description S of a model of Σ together with a set U of conditions guaranteeing the "executability" of S. Moreover, we propose an extension of the standard tableau calculus for LTL so that a conditioned model of a set Σ of formulae can be extracted from any complete and open branch of a tableau for Σ. As can be expected, the set U of conditions extracted from the analysis of a single branch is not necessarily minimal. In order to prune U from redundant conditions, the whole tableau must be built and each of its open branches possibly expanded further.

When not only a single conditioned plan is looked for, but a whole set of complementary ones, the construction of a single tableau branch is not sufficient in any case. We notice, in this respect, that the theorem prover `ptl` [5] used by the planning system described in [2] (and whose comparison with other existing planning systems has given encouraging results) builds the whole graph representing a tableau, before a single branch is analysed in order to extract a model. Conditional planning can therefore be accomplished by means of a non-substantial modification of the algorithm underlying `ptl`.

As a further remark, we observe that the requirement for minimality can reasonably be relaxed when a single conditioned model is to be used because planning and execution (with sensing actions) are intermingled. On the other hand, some preliminary experiments with a simple prototype, implementing the system described in this work, have always produced a minimal set $U(K)$ explaining $lits(K)$ in the context of the initial set of formulae.

Note, finally, that the same machinery can deal with *conformant planning* [15], i.e. the search for non-conditional plans succeeding no matter which of the allowed states the world is actually in. A conformant plan in fact corresponds to a conditioned plan with empty conditions.

Directions for future work include:

- the definition of a calculus aiming at producing *finite* temporal models of the initial set of formulae, in view of the fact that a plan consists of a finite sequence of actions. The calculus would be simplified in this case by the absence of open looping branches (by Lemma 4.5 in [14], such models can be discarded). Possibly, also the computation of a minimal set of conditions is simplified.
- The study of reasonable rules acting on the C-part of cl-formulae and allowing one to restrict the set of conditions associated with a branch.
- The analysis of the syntactical form of the encoding of planning problems, and its possible relations with minimality.

Acknowledgements. The authors thank the anonymous referees of this work for their precious comments and suggestions.

References

1. S. Cerrito and M. Cialdea Mayer. Using linear temporal logic to model and solve planning problems. In F. Giunghiglia, editor, *Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, pages 141–152. Springer, 1998.
2. M. Cialdea Mayer, A. Orlandini, G. Balestreri, and C. Limongelli. A planner fully based on linear time logic. In S. Chien, S. Kambhampati, and C.A. Knoblock, editors, *Proc. of the 5th Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, pages 347–354. AAAI Press, 2000.
3. A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *Proc. of the International Conference on AI Planning and Scheduling (AIPS'98)*. AAAI Press, 1998.
4. A. Finzi, F. Pirri, and R. Reiter. Open world planning in the Situation Calculus. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 754–760. AAAI Press, 2000.
5. G. L. J. M. Janssen. *Logics for Digital Circuit Verification. Theory, Algorithms and Applications.* CIP-DATA Library Technische Universiteit Eindhoven, 1999.
6. H. Kautz and B. Selman. Planning as satisfiability. In B. Neumann, editor, *10th European Conference on Artificial Intelligence (ECAI)*, pages 360–363. Wiley & Sons, 1992.
7. H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 1202–1207. AAAI Press / The MIT Press, 1996.
8. H. Levesque. What is planning in the presence of sensing? In *Proc. of the 13th National Conference on Artificial Intelligence, AAAI-96*, pages 1139–1146. AAAI Press, 1996.
9. M. A. Peot and D. E. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Proc. of the First International Conference on Artificial Intelligence Planning Systems*, pages 189–197. Morgan Kaufmann, 1992.
10. M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proc. IJCAI'01*. AAAI Press, 2001.
11. L. Pryor and G. Collins. Planning for contingencies: a decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.
12. R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and mathematical theory of computation: Papers in honor of John McCarthy*, pages 359–380. Academic Press, 1991.
13. J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intellingence Research*, 10, 1999.
14. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
15. D. E. Smith and D. S. Weld. Conformant Graphplan. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 889–896, 1998.
16. P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28:119–152, 1985.