TabPlan: Planning in Linear Temporal Logic

Rapport de Recherche L.R.I.

Université de Paris Sud
Centre d'Orsay
Laboratoire de Recherche en Informatique
Bât 490
91405 Orsay (France)

Serenella Cerrito

Université de Paris-Sud,
LRI, Bât 490,
F-91405 Orsay Cedex, France.
e-mail: serena@lri.fr
Tel. +33-1-69154244

Marta Cialdea Mayer*

Università di Roma Tre,
Dip. di Informatica e Automazione,
Via Vasca Navale 84, 00146 Roma, Italia.
e-mail: cialdea@inf.uniroma3.it
Tel. +39-6-55177054

*This work has been partially supported by MURST, CNR and ASI.

Abstract

In this work a tableau calculus is proposed, that checks whether a finite set of formulae in propositional linear temporal logic (LTL) has a finite model whose cardinality is bounded by a constant given in input, and constructs such a model, if any. From a theoretical standpoint, the method can also be used to check finite satisfiability *tout court*. The following properties of the proposed calculus are proved: termination, soundness and completeness w.r.t. finite model construction.

The motivation behind this work is the design of a logical language to model planning problems and an associated calculus for plan construction, integrating the declarativity, expressiveness and flexibility typical of the logical languages with the capability of embedding search-based techniques well established in the planning community.

Résumé

Nous proposons un calcul par tableaux, qui teste si un ensemble fini de formules propositionnelles de la logique temporelle linéaire (LTL) a un modèle fini dont la taille est bornée par une constante donnée en entrée et, si ceci est le cas, construit un tel modèle. Du point de vue théorique, cette méthode peut être aussi utilisée pour tester la satisfiabilité finite, tout court. Les propriétés suivantes du calcul proposé sont démontrées: terminaison, correction et complétude par rapport à la construction des modèles finis.

La motivation de ce travail est la définition d'un langage logique apte à modéliser des problèmes de planification et d'un calcul correspondant pour la construction de plans, intégrant la declarativité, l'expressivité et la flexibilité typiques des langages logiques avec la capacité d'incorporer des techniques de recherche bien établies dans la communauté de la planification.

1 Introduction

This work investigates the issue of the construction of *finite* models of a given set of linear temporal formulae and, in particular, its application to model and solve planning problems. Model construction in temporal logic is the main concern of *executable temporal logics*, that are based on an imperative reading of logic, where executing a temporal specification means constructing a model for it [FO95]. However, the temporal model underlying executable temporal logics is infinite in the future, their main original motivation being the specification and verification of reactive systems.

In [KS92] the view of a planning activity is proposed, as the search for a finite model of the specification of the planning problem. In such a perspective, planning amounts to prove that $preconditions \wedge goals$ has a finite model and such a model represents, in fact, a plan achieving the desired goals.

Except for this and few other proposals (e.g. [BK96b, BK96a]), the logical approach to planning is usually based on deduction, and plan generation is carried out by constructively *proving* plan specification formulae, having the form $preconditions \rightarrow goals$. Within this approach, different logics have been used: classical logic [Rei91], linear logic [Mas93, MTV93], temporal logics [SB96, KT95]. Works using modal temporal logics rely on the branching model of time of interval-based temporal logics.

The other main approach to planning, deriving from classical STRIPS, is based on *ad hoc*, more or less powerful formalisms, and solves planning problems by means of search based techniques (see, for example, [CO96, CT91, SPW92]). Some of the planning languages in this category have a definitely unclear semantics. The problem specification sometimes codes procedural information, used to guide the search, that leads the planner to prune part of the potential search space, thus gaining in efficiency but risking to lose completeness.

Other planning formalisms have been given a clean semantics, but they generally have a limited expressive power, corresponding to significant restrictions of first order logic. The problem of designing more expressive and *easy-to-use* planning languages, with a clear and natural semantics, is felt as an important one in the planning community.

On the other hand, the use of specialized algorithms and techniques, such as partial order, regression planning, least commitment (see Section 5), as well as control strategies to guide the search, may overperform the use of general theorem provers as tools for plan construction. In fact, in many planning problems, methods based on regression and plan space search lead to a significant reduction of the search space.

The motivation of this work is the design of a logical language and an associated calculus for plan construction having the following features:

- the declarativity, expressiveness and flexibility typical of the logical approach, together with a clean semantics based on a natural model of time and actions;

- the capability of embedding search-based techniques, typically partial order, regression and least committment.

Viewing a plan as a finite model of its specification, this work proposes the use of linear temporal logic (LTL) as the specification language (differently, [KS92] use propositional classical logic). The main advantage of the use of LTL as a planning language derives from its rich expressive power and the underlying simple model of time and actions.

The search for finite LTL models is carried out by means of tableau construction. Semantical tableau bear in fact a significant advantage with respect to other proof systems strongly based on normal forms (resolution oriented, for instance) when proofs are expected to convey understandable information to the user. This is in fact the case in the planning domain, since it is widely recognized that real planning problems can hardly be solved in a complete automatic way, thus making it preferable the design of mixed-initiative systems. Moreover, tableau methods have proven to be efficient reasoning methods for practical purposes when the proof search is guided by control knowledge that help keep the search space to a manageable size.

The tableau calculus described in this work carries strong similarities with [SGL97, SGL]. However, while temporal structures are alway infinite in these works, we focus on finite temporal models, that are obviously more suitable to represent plans. Moreover, since planning is a notoriusly hard problem and practical solutions are generally required not to exceed a given (polynomial) "length" known in advance (*bounded* planning problems), we focus on the search of finite models whose size is not greater than a given limit, even if, from a theoretical standpoint, the calculus could be used also to check finite satisfiability *tout court*. As a byproduct of the focus on finite models, the "since" and "until" operators can be given an intuitive and simple treatment, without hindering termination. As already remarked, the naturalness of the expansion rules is an important feature for the intended application.

We stress that in this work we focus on the theoretical grounds of the integration of search-based techniques in an LTL tableau calculus, without specifically addressing efficiency issues. Practical systems for model construction can be built on this basis, by studying meaningful subclasses of the language, allowing the encoding of planning problems. These two phases are typical of the development of logical approaches to AI problems.

The rest of this paper introduces the syntax and finite model semantics of propositional linear temporal logic (Section 2) and the prefixed tableau calculus for it (Section 3). Soundness, completeness w.r.t. finite model construction and decidability are established (Section 4). Section 5 briefly illustrates some techniques used in search-based planning and reflected by the tableau calculus, and describes some features of a prototype implementation of the system. Section 6 concludes this work.

2 Linear Temporal Logic

The language of linear temporal logic we consider extends classical propositional logic, with the logical operators $\neg, \wedge, \vee$, $\bot$ (always false) and $\top$ (always true), by means of the unary modal operators $\Box$ (always in the future), $\overleftarrow{\Box}$ (always in the past), $\Diamond$ (sometime in the future), $\overleftarrow{\Diamond}$ (sometime in the past), and the binary ones $\mathcal{S}$ (since) and $\mathcal{U}$ (until).

The semantics of the language is defined as follows. A *temporal structure* T is a finite initial segment of the natural numbers: $\langle 0, ..., k\rangle$; its elements are called *time points*. If $\mathcal{L}$ is an LTL temporal language and P the set of propositional letters in $\mathcal{L}$, an $\mathcal{L}$*-interpretation* $\mathcal{M}$ is a pair $\langle T, \sigma\rangle$, where:

- T is a temporal structure;
- $\sigma : T \to \mathcal{P}(P)$ is a function on time points, providing an interpretation to the propositional letters in P for any point in T. I.e. if $i \in T$, then $\sigma(i) \subseteq P$ is the set of propositional letters true in i.

The satisfiability relation $\mathcal{M}_i \models A$, for $i \in T$, is inductively defined as follows:

1. $\mathcal{M}_i \models \top$
2. $\mathcal{M}_i \not\models \bot$
3. $\mathcal{M}_i \models p$ iff $p \in \sigma(i)$, for $p \in P$.
4. $\mathcal{M}_i \models \neg p$ iff $p \notin \sigma(i)$, for $p \in P$.
5. $\mathcal{M}_i \models A \wedge B$ iff $\mathcal{M}_i \models A$ and $\mathcal{M}_i \models B$.
6. $\mathcal{M}_i \models A \vee B$ iff either $\mathcal{M}_i \models A$ or $\mathcal{M}_i \models B$.
7. $\mathcal{M}_i \models \Box A$ iff for all $j > i$, $\mathcal{M}_j \models A$.
8. $\mathcal{M}_i \models \Diamond A$ iff there exists $j \in T$ such that $j > i$ and $\mathcal{M}_j \models A$.
9. $\mathcal{M}_i \models A\mathcal{U}B$ iff there exists $j \in T$ such that $j > i$ and $\mathcal{M}_j \models B$ and for any k with $i < k < j$ $\mathcal{M}_{||} \models \mathcal{A}$.
10. $\mathcal{M}_i \models \overleftarrow{\Box} A$ iff for all $j < i$, $\mathcal{M}_j \models A$.
11. $\mathcal{M}_i \models \overleftarrow{\Diamond} A$ iff there exists $j \in T$ such that $j < s$ and $\mathcal{M}_j \models A$.
12. $\mathcal{M}_i \models A\mathcal{S}B$ iff there exists $j \in T$ such that $j < i$ and $\mathcal{M}_j \models B$ and for any k with $i < k < j$ $\mathcal{M}_k \models A$.

Other temporal operators can be defined:

$$
\begin{array}{ll}
A\mathcal{W}B \equiv_{def} (A\mathcal{U}B) \vee \Box A & A\mathcal{Z}B \equiv_{def} (A\mathcal{S}B) \vee \overleftarrow{\Box} A \\
\bar{\Box} A \equiv_{def} A \wedge \Box A & \bar{\Diamond} A \equiv_{def} A \vee \Diamond A
\end{array}
$$

And, due to the "strong" interpretation of the modal operators (excluding the present time point also in the case of future time operators), the weak and strong "Next" and "Last" operators are also definable:

$$
\begin{array}{ll}
\odot A \equiv_{def} \bot \mathcal{U} A & \overleftarrow{\odot} A \equiv_{def} \bot \mathcal{S} A \\
\bigcirc A \equiv_{def} \bot \mathcal{W} A & \overleftarrow{\bigcirc} A \equiv_{def} \bot \mathcal{Z} A
\end{array}
$$

Although also the $\Diamond$ and $\overleftarrow{\Diamond}$ operators are definable, we prefer to give them a separate treatement in the tableau systems.

Truth is defined as satisfiability in the initial state: a formula A is true in $\mathcal{M}$ iff $\mathcal{M}_0 \models A$, i.e. A is satisfied in the first state 0. If $\mathcal{M} \models_0 A$ then $\mathcal{M}$ is a model of A. Truth of sets of formulae is defined as usual. Two formulae A and B are equivalent iff for all $\mathcal{M}$ and i, $M_i \models A$ iff $\mathcal{M}_i \models B$.

A formula is in *negation normal form* (nnf) iff no logical operator is in the scope of a negation. It can easily be shown that, under the strong notion of equivalence above introduced, every formula can be transformed into an equivalent formula in negation normal form (nnf), by applying the usual rules for $\rightarrow, \wedge, \vee, \Box, \overleftarrow{\Box}, \Diamond, \overleftarrow{\Diamond}$ and:

$$
\neg(A\mathcal{S}B) \equiv (\neg B\mathcal{S}(\neg A \wedge \neg B)) \vee \overleftarrow{\Box}\neg B
$$

$$
\neg(A\mathcal{U}B) \equiv (\neg B\mathcal{U}(\neg A \wedge \neg B)) \vee \Box\neg B
$$

3 Tableaux

In this section a tableau system is described, that allows one to test whether a set of LTL formulae in nnf admits models whose underlying temporal structure does not exceed a given size.[1] In case of a positive answer, a model can be easily extracted from the tableau construction.

Let $C = \{start, finish, d_1, d_2, d_3, ...\}$ be a set of *constants* (intuitively denoting time points). Then:

- A *state* is any expression of the form $a + n$, for $a \in C$ and $n \in Z$. The set of states is denoted by Σ. It is intended that $C \subset \Sigma$ (a can be rewritten as $a + 0$).
- If $s, t \in \Sigma$, then $s \leq t$ is a *temporal constraint*.

[1] The restriction to nnf allows for a more compact description of the system.

- A *labelled formula* an expression of the form $[s,t]A$, where $s,t \in \Sigma$ and A is an LTL formula in negation normal form. $[s,s]A$ will be abbreviated by $[s]A$.

A tableau is a tree whose nodes are labelled either by temporal constraints or labelled formulae (that are called *logical nodes*). A branch in a tableau $\mathcal{T}$ is a maximal path in $\mathcal{T}$, i.e. a sequence of nodes from the root of $\mathcal{T}$ to a leaf.

The expansion rules of the calculus are in Table 1, where a, b denote elements of C, $s, t, s', t', ..$ elements of Σ. The set of nodes occurring above the line of a rule is called the *premisse* of the rule, while the sets of nodes occurring below are the *expansions* of the premisse. Non-branching rules have a single expansion, and branching rules have two expansions.

Tableaux are inductively defined as follows. Let S be a finite set of formulae in nnf and K the set $\{finish \leq start + k\}$ for some integer $k \geq 0$ (the role of K is the specification of the maximal size of allowed models). Then:

- An initial tableau for $S \cup K$ is a one-branch tableau whose nodes are labelled by $\{[start]A \mid A \in S\} \cup K$
- if $\mathcal{T}$ is a tableau for $S \cup K$, $\mathcal{B}$ a branch in $\mathcal{T}$ and $X_1, .., X_n$ nodes in $\mathcal{B}$ that match the premise of an expansion rule $\mathcal{R}$ in Table 1, then $\mathcal{T}$ can be expanded to a tableau $\mathcal{T}'$ for $S \cup K$ (by application of $\mathcal{R}$ to $X_1, .., X_n$) by adding the expansion(s) of $\mathcal{R}$ to $\mathcal{B}$'s leaf.

Note that the β-rule contains a sort of contraction: the rightmost expansion of the rule contains a node with the same formula already occurring in the premise, even though the labels (intervals) of the nodes are different.

When the interval rule is applied to expand $[s,t]A$, we say that it is applied to the interval $[s,t]$, independently of the formula A. This rule could be dispensed with, and the corresponding branching added to most of the other rules. Its distinction as a separate rule makes the formulation of the calculus more compact.

When the leftmost conflict resolution rule is applied, we say that it is applied to the nodes $[s,t]\,p$ and $[s',t']\,\neg p$.

In the following, if $\mathcal{B}$ is a tableau branch, $const(\mathcal{B})$ denotes the set of constants occurring in $\mathcal{B}$ and including $start$ and $finish$.

Definition 1 *Let C be a set of constants (including start and finish) and $\mathcal{I}$ a mapping from C to the integers. The notation $\mathcal{I}^*$ is used to denote the extension of $\mathcal{I}$ from states to the integers such that $\mathcal{I}^*(a+n) = \mathcal{I}(a) + n$ for every $a \in C, n \in Z$.*

1. *Let $T = \langle 0, ..., k\rangle$ be a finite sequence of integers starting at 0. $\mathcal{I}$ is a* temporal mapping *for C with range T if*

   (a) $min\{\mathcal{I}(a) \mid a \in C\} = \mathcal{I}(start) = 0$;

Interval rule
$\dfrac{[s,t]\,A}{t \leq s-1 \qquad\qquad s \leq t}$

Logical rules		
α-rule	**β-rule**	
$\dfrac{\begin{array}{c}[s,t]\,A \wedge B\\ s \leq t\end{array}}{\begin{array}{c}[s,t]\,A\\ [s,t]\,B\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A \vee B\\ s \leq t\end{array}}{[s,t]\,A \qquad \begin{array}{c}s \leq a\\ a \leq t\\ [s,a-1]\,A\\ [a]\,B\\ [a+1,t]\,A \vee B\\ a \in C \text{ fresh}\end{array}}$	
$\Box$-rule	**$\Diamond$-rule**	**$\mathcal{U}$-rule**
$\dfrac{\begin{array}{c}[s,t]\,\Box A\\ s \leq t\end{array}}{[s+1, finish]\,A}$	$\dfrac{\begin{array}{c}[s,t]\,\Diamond A\\ s \leq t\end{array}}{\begin{array}{c}[b]\,A\\ t+1 \leq b\\ b \in C\ fresh\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A\mathcal{U}B\\ s \leq t\end{array}}{\begin{array}{c}[a]\,B\\ t+1 \leq a\\ [t+1,a-1]\,A\\ [s+1,t]\,A \vee B\\ a \in C\ fresh\end{array}}$
$\overleftarrow{\Box}$-rule	**$\overleftarrow{\Diamond}$-rule**	**$\mathcal{S}$-rule**
$\dfrac{\begin{array}{c}[s,t]\,\overleftarrow{\Box}A\\ s \leq t\end{array}}{[start, t-1]\,A}$	$\dfrac{\begin{array}{c}[s,t]\,\overleftarrow{\Diamond}A\\ s \leq t\end{array}}{\begin{array}{c}[b]\,A\\ b \leq s-1\\ b \in C\ fresh\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A\mathcal{S}B\\ s \leq t\end{array}}{\begin{array}{c}[a]\,B\\ a \leq s-1\\ [a+1,s-1]\,A\\ [s,t-1]\,A \vee B\\ a \in C\ fresh\end{array}}$

Conflict resolution rules	
$\dfrac{\begin{array}{c}s \leq t\\ s' \leq t'\\ [s,t]\,p\\ [s',t']\,\neg p\end{array}}{t \leq s'-1 \qquad\qquad t' \leq s-1}$	$\dfrac{[s,t]\,\bot}{t \leq s-1}$

Table 1: Tableau expansion rules

   (b) $max\{\mathcal{I}(a) \mid a \in C\} = \mathcal{I}(finish) = k$.

   Hence, in particular, the range of a temporal mapping is always finite.

2. *If K is a set of temporal constraints over C, then $\mathcal{I}$ is a solution to K iff:*

   (a) *$\mathcal{I}$ is a temporal mapping for C;*

   (b) *if $s \leq t \in K$, then $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$.*

3. *Let $\mathcal{B}$ be a tableau branch, $C = const(\mathcal{B})$ and $\mathcal{M}$ a temporal interpretation with domain T.*

   (a) *If $\mathcal{I}$ is a temporal mapping for C with range T, then $\langle \mathcal{M}, \mathcal{I} \rangle$ satisfies $\mathcal{B}$ ($\langle \mathcal{M}, \mathcal{I} \rangle \models \mathcal{B}$) iff:*

      i. *$\mathcal{I}$ is a solution to the set of temporal constraints occurring in $\mathcal{B}$;*

      ii. *if $[s,t]A$ occurs in $\mathcal{B}$, then for every integer i, if $i \in T$ and $I^*(s) \leq i \leq I^*(t)$, then $\mathcal{M}_i \models A$.*

   (b) *$\mathcal{B}$ is* satisfiable *in $\mathcal{M}$ iff there exists a temporal mapping $\mathcal{I}$ for C such that $\langle \mathcal{M}, \mathcal{I} \rangle \models \mathcal{B}$.*

Definition 2 *Let $\mathcal{B}$ be a tableau branch and K the set of temporal constraints occurring in $\mathcal{B}$. $\mathcal{B}$ is* open *iff there exists a solution to K. Otherwise it is* closed.

The following definition captures the intuitive idea of tableaux where no wasteful expansions are ever performed. In particular, closed branches are never expanded.

Definition 3 *A tableau branch $\mathcal{B}$ is canonical iff:*

- *The interval rule is applied no more than once to each interval.*
- *Every labelled formula is expanded no more than once by means of a logical rule.*
- *The conflict resolution rule is applied no more than once to each node or pair of nodes.*
- *No proper initial subsegment of $\mathcal{B}$ is closed.*

A tableau is canonical iff all its branches are canonical.

Definition 4 *If $\mathcal{B}$ is a tableau branch, then $\mathcal{B}$ is* complete *iff there exists no canonical expansion of $\mathcal{B}$. A tableau is complete if all its branches are complete.*

4 Termination, Completeness and Soundness

In this section we show that the construction of any canonical tableau terminates and that the calculus is complete and sound, i.e., if a given set of formulae S has a model $\mathcal{M}$ whose size is not greater than k, then any tableau for $S \cup \{finish \leq start + k\}$ has an open branch that is satisfiable in $\mathcal{M}$ (completeness) and any complete and open branch, in any tableau for $S \cup \{finish \leq start + k\}$, is satisfiable in some model whose size is bounded by k (soundness). It is also shown that, from a theoretical standpoint, the calculus can be used to check satisfiability in models of any finite size.

4.1 Termination

Here we restrict our attention to canonical tableaux, thus ensuring, in particular, that the construction of a branch is abandoned as soon as it is recognized to be closed. In this respect, it must be noted that the existence of a solution for a given set of constraints can be tested by means of well-known graph algorithms (see, for example, Chapter 25 in [CLR90]).

Let $K = \{finish \leq start + n\}$, where n is any natural number. Consider the following (non-deterministic) procedure of tableau construction:

- The tableau is initialized to $[start]F$ and K.
- Nodes are systematically expanded by means of the rules of the calculus. Every time a new temporal constraint is added to the branch, it is tested whether it is still open; if it is not open, the expansion is halted.

Below, we show that, although branch construction might continue ad libitum if K were empty, it always terminates if K has the form $\{finish \leq start + n\}$.

The following definition captures the idea of a sequence of applications of the β-rule, each of which expands an expansion of the previous one.

Definition 5 *Let $\mathcal{B}$ be a tableau branch. A β-node in $\mathcal{B}$ is a node of the form $[s,t]A \vee B$. A β-chain in $\mathcal{B}$ is a sequence of β-nodes $X_0, X_1, \ldots$ such that, for every $i \geq 1$, X_{i-1} is expanded in $\mathcal{B}$ by application of the β-rule and X_i is the β-node in the corresponding rightmost expansion. A k-length-β-chain is a finite β-chain $X_0, X_1, \ldots, X_{k+1}$, constituted by $k+2$ nodes.*

A node X in $\mathcal{B}$ is the root node *of a β-chain if it is the first node in a β-chain and it is not itself obtained by an application of the β-rule, i.e. there exists a maximal length β-chain in $\mathcal{B}$ having X as its first node.*

The proof that the construction of canonical tableaux terminates uses the following lemmas.

Lemma 1 *Let $\mathcal{B}$ be a branch in a tableau and $\mathcal{I}$ a temporal mapping satisfying the constraints in $\mathcal{B}$. If $[s,t]A$ occurs in $\mathcal{B}$ and $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$, then $0 \leq I^*(s) \leq I^*(t) \leq \mathcal{I}(finish)$.*

Proof. The lemma is obviously satisfied by the logical nodes in the initial one-branch tableau, that have the form $[start]A$. Hence, what has to be shown is that, for any application of an expansion rule, if every node of the form $[s,t]A$ in the premisse satisfies $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$ whenever $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$, then every node in the expansion(s) having the the form $[s',t']B$ is such that, if $\mathcal{I}^*(s') \leq \mathcal{I}^*(t')$, then $0 \leq \mathcal{I}^*(s') \leq \mathcal{I}^*(t') \leq \mathcal{I}(finish)$.

The interval rule, the α-rule and the conflict resolution rules do not introduce any new interval, hence the corresponding cases are straightforward. The same holds for the $\Diamond$ and $\overleftarrow{\Diamond}$ rule, that only introduce intervals of the form $[a,a]$, where a is a constant. In fact $0 \leq \mathcal{I}(a) \leq \mathcal{I}(finish)$ holds if $\mathcal{I}$ is a temporal mapping for the constants occurring in $\mathcal{B}$. The other cases are treated as follows.

1. Let us assume that the β-rule is applied to expand $[s,t]A \vee B$, and that $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$. Whichever the value of $\mathcal{I}(a)$, if $\mathcal{I}$ satisfies the constraints in $\mathcal{B}$, it holds that $s \leq a \leq t$, hence $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}(a) \leq \mathcal{I}(finish)$. Moreover, $0 \leq \mathcal{I}^*(a+1)$; hence if $\mathcal{I}^*(a+1) \leq \mathcal{I}^*(t)$, then $0 \leq \mathcal{I}^*(a+1) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$ holds too.

2. In the case of the $\overleftarrow{\Box}$-rule, by hypothesis $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$. Then clearly $\mathcal{I}^*(t-1) \leq \mathcal{I}(finish)$. On the other hand, it must be shown that $0 \leq \mathcal{I}^*(t-1)$ only if $\mathcal{I}(start) \leq \mathcal{I}^*(t-1)$, that is straightforward. The fact that $0 \leq \mathcal{I}(start) \leq \mathcal{I}(finish)$ is straightforward too.

3. Let the $\mathcal{S}$-rule be applied to $[s,t]A\mathcal{S}B$, with $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$, producing $a \leq s-1$, $[a]B$, $[a+1,s-1]A$ and $[s,t-1]A \vee B$. The interval $[a]$ trivially satisfies the desired property.

   Moreover, $0 \leq \mathcal{I}^*(a+1)$ and $\mathcal{I}^*(s-1) \leq \mathcal{I}(finish)$, hence if also $\mathcal{I}(a+1) \leq \mathcal{I}^*(s-1)$, then $0 \leq \mathcal{I}^*(a+1) \leq \mathcal{I}^*(s-1) \leq \mathcal{I}(finish)$.

   Similarly, $0 \leq \mathcal{I}^*(s)$ and $\mathcal{I}^*(t-1) \leq \mathcal{I}(finish)$, hence if also $\mathcal{I}(s) \leq \mathcal{I}^*(t-1)$, then $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t-1) \leq \mathcal{I}(finish)$.

The $\Box$ and $\mathcal{U}$-rules are treated similarly. ∎

In the following, we use the term "subformula" so that $A \vee B$ is included among the subformulae of both $A\mathcal{U}B$ and $A\mathcal{S}B$. Given a formula F and a set of formulae E, the notations $sub(F)$ and $sub(E)$ denote, respectively, the set of subformulae of F and the set of subformulae of the formulae in E.

Lemma 2

1. *If $\mathcal{B}$ is a canonical open branch in a tableau for $S \cup \{finish \leq start + n\}$, then $\mathcal{B}$ contains no k-length-β-chain with $k > n$.*

2. *Any canonical infinite branch contains an infinite β-chain.*

3. *Every canonical open branch is finite.*

Proof.

1. Let us assume that $\mathcal{B}$ does contain a k-length-β-chain with $k > n$. Then $\mathcal{B}$ contains a sequence of nodes having the form: $[s,t]\ A \vee B, [a_1+1,t]\ A \vee B, [a_2+1,t]\ A \vee B, ..., [a_{n+2}+1,t]\ A \vee B$ and the sequence of constraints: $s \leq a_1, a_1 + 1 \leq a_2, ..., a_{n+1} + 1 \leq a_{n+2}$, and $a_{n+2} \leq t$. If $\mathcal{I}$ is is a solution for the constraints in $\mathcal{B}$, then, by Lemma 1, $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}^*(finish)$ (in fact $s \leq t$ is in $\mathcal{B}$), so that $\mathcal{I}^*(a_{n+2}) \geq n+1$. However, $\mathcal{I}^*(a_{n+2}) \leq \mathcal{I}^*(t) \leq \mathcal{I}^*(finish) \leq n$ holds too, that is absurd.

2. It is easily recognized that the calculus enjoys the subformula property: if $[s,t]A$ occurs in the premisse of a rule, then the (only) logical node in any of its expansions has the form $[s',t']B$, where A is a (not necessarily proper) subformula of B.

   Hence, if a canonical branch $\mathcal{B}$ is infinite, then there is an infinite number of nodes X_i in $\mathcal{B}$ having the form $[s_i,t_i]A$, for a given formula A. Cleary, A must be a disjunction. Let S be the set of those X_i that are expanded in $\mathcal{B}$, so that every node in S belongs to some β-chain in $\mathcal{B}$. Clearly, S is still infinite. The set S can be partitioned into classes $C_1, C_2, ...$ such that X_i and X_j belong to the same class if and only if the maximal β-chain to which they belong has the same root node. Hence, X_i and X_j belong to the same class iff they belong to the same β-chain.

   We consider two cases:

   - Every class C_i is finite, hence S is partitioned into an infinite number of classes. Therefore there is an infinite number of root nodes of β-chains in $\mathcal{B}$. Each of such nodes R_i is the expansion of a node Y_i that is not a β-node. Hence, some rule other than the β-rule is applied infinitely often in $\mathcal{B}$. But this is absurd, since the initial tableau is always finite and $\mathcal{B}$ is canonical, so that the same formula cannot be expanded an infinite number of times. Therefore, only the second case is possible:
   - There is a class C_i containing an infinite number of nodes. In this case, an infinite number of β-nodes of $\mathcal{B}$, namely the elements of C_i, belong to the same maximal β-chain, that is therefore infinite.

3. Let $\mathcal{B}$ be a canonical open branch. Then the set of its temporal constraints has a solution $\mathcal{I}$, hence, by item 1, $\mathcal{B}$ contains no infinite β-chain and, by item 2, it is finite.

$\blacksquare$

From the above lemma, it immediately follows that:

Theorem 1 *Any canonical tableau for $S \cup \{finish \leq start + n\}$, for any $n \geq 0$, is finite.*

4.2 Completeness with respect to model construction

The proof that the tableau calculus is complete w.r.t. model construction uses the following lemmas.

Lemma 3 *Let $\mathcal{B}$ be a branch in a tableau $\mathcal{T}$, $\mathcal{M}$ a temporal interpretation and $\mathcal{I}$ a temporal mapping for the constants occurring in $\mathcal{B}$. If $\langle \mathcal{M}, \mathcal{I} \rangle \models \mathcal{B}$ and $\mathcal{B}$ can be expanded, $\mathcal{B}_1$ (and $\mathcal{B}_2$) being its expansion(s), then for some $i = 1, 2$ there exists an extention $\mathcal{I}'$ of $\mathcal{I}$ to the fresh constants of $\mathcal{B}_i$ (if any) such that $\langle \mathcal{M}, \mathcal{I}' \rangle \models \mathcal{B}_i$*

Proof. Different cases must be considered according to the applied expansion rule. Let T be the temporal structure underlying $\mathcal{M}$ (and the range of $\mathcal{I}$).

1. The cases of the interval rule and the α-rule are straightforward.

2. In the case of the β-rule, by hypothesis $\langle \mathcal{M}, \mathcal{I} \rangle \models [s, t]A \vee B$ and $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$, so that $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}^*(finish)$, by Lemma 1. It follows that for every $i \in [\mathcal{I}^*(s), \mathcal{I}^*(t)]$, $\mathcal{M}_i \models A \vee B$. If for every such i, $\mathcal{M}_i \models A$, then the leftmost branch is clearly satisfied by $\mathcal{M}$ and $\mathcal{I}$. Otherwise, let k be the smallest element in the interval $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$ such that $\mathcal{M}_k \not\models A$, hence $\mathcal{M}_k \models B$. The mapping $\mathcal{I}$ is extended to $\mathcal{I}'$ such that $\mathcal{I}'(a) = k$. By the choice of k, clearly $\langle \mathcal{M}, \mathcal{I}' \rangle \models [s, a-1]A$ and $\langle \mathcal{M}, \mathcal{I}' \rangle \models [a]B$. If $k = \mathcal{I}^*(t)$, $\langle \mathcal{M}, \mathcal{I}' \rangle \models [a+1, t]A \vee B$ is voidly satisfied. Otherwise, the interval $[\mathcal{I}^*(a+1), \mathcal{I}^*(t)]$ is a subinterval of $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$, hence again $\langle \mathcal{M}, \mathcal{I}' \rangle \models [a+1, t]A \vee B$.

3. In the case of the $\overleftarrow{\Box}$-rule, by hypothesis $\langle \mathcal{M}, \mathcal{I} \rangle \models [s, t]\overleftarrow{\Box}A$ and $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$. Hence for every $i \in T$ such that $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$, $\mathcal{M}_i \models \overleftarrow{\Box}A$. Since $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$ (by Lemma 1), $\mathcal{I}^*(t) \in T$ and $\mathcal{I}^*(t) \in [\mathcal{I}^*(s), \mathcal{I}^*(t)]$. Hence $\mathcal{M}_{\mathcal{I}^*(t)} \models \overleftarrow{\Box}A$, i.e. for every $i \in T$ such that $i < \mathcal{I}^*(t)$, $\mathcal{M}_i \models A$. This clearly implies that $\langle \mathcal{M}, \mathcal{I} \rangle \models [start, t-1]A$.

4. In the case of the $\overleftarrow{\Diamond}$ rule, applied to a node $[s, t]\overleftarrow{\Diamond}A$, by hypothesis we have that $\langle \mathcal{M}, \mathcal{I} \rangle \models [s, t]\overleftarrow{\Diamond}A$ and $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$. By Lemma 1, $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$, hence $\mathcal{I}^*(s) \in T$. Therefore there exists $i \in T$ s.t. $i < \mathcal{I}^*(s)$ and $\mathcal{M}_i \models A$. If b is the new constant introduced by the rule, by setting $\mathcal{I}'(b) = i$ we get the desired conclusion.

5. Let the $\mathcal{S}$-rule be applied to $[s,t]A\mathcal{S}B$. By hypothesis, $\langle \mathcal{M}, \mathcal{I}\rangle \models A\mathcal{S}B$ and $\mathcal{I}(s) \leq \mathcal{I}^*(t)$. By Lemma 1, $0 \leq \mathcal{I}^*(s) \leq \mathcal{I}^*(t) \leq \mathcal{I}(finish)$. It follows that $\mathcal{I}^*(s) \in T$ and that there exists $i \in T$ such that $i < \mathcal{I}^*(s)$, $\mathcal{M}_i \models B$, and for any $j \in T$ such that $i < j < \mathcal{I}^*(s)$, $\mathcal{M}_j \models A$. Hence if we set $\mathcal{I}'(a) = i$, where a is the new constant introduced by the rule, we get that $\langle \mathcal{M}, \mathcal{I}'\rangle$ satisfies $[a]B$, $a \leq s-1$ and $[a+1, s-1]A$. Let now consider any time point $j \in T$ such that $\mathcal{I}^*(s) \leq j \leq \mathcal{I}^*(t-1)$ and assume that $\mathcal{M}_j \not\models A \vee B$. Let k be the smallest of such points; then $\mathcal{I}^*(s) \leq k+1 \leq \mathcal{I}^*(t)$ but $\mathcal{M}_{k+1} \not\models A\mathcal{S}B$, a contradiction.

6. The treatment of the other logical rules is symmetrical.

7. The case of the conflict resolution rules is easily settled by observing that, under the hypotheses of the lemma, it must be the case that the intervals $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$ and $[\mathcal{I}^*(s'), \mathcal{I}^*(t')]$ have an empty intersection (first rule), and that the interval $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$ is empty (second rule).

■

As a consequence of the above lemma and Theorem 1 we have the following theorem.

Theorem 2 *Let $\mathcal{M} = \langle T, \sigma\rangle$ be a temporal interpretation, with $T = \langle 0, ..., n\rangle$, S a finite set of LTL formulae in nnf, and $K = \{finish \leq start + k\}$, for some integer $k \geq n$. If $\mathcal{M}$ is a model of S then any canonical tableau for $S \cup K$ has a branch that is finite, open and satisfiable in $\mathcal{M}$.*

Proof. If $\mathcal{B}_{init}$ is the initial tableau for $S \cup K$, then, obviously $\langle \mathcal{M}, \mathcal{I}\rangle \models \mathcal{B}_{init}$, where $\mathcal{I}$ is the temporal mapping for $\{start, finish\}$ with range T. By repeated applications of Lemma 3, there exists a branch that is satisfiable in $\mathcal{M}$; such a branch is obviously open and, by Lemma 2, it is finite. ■

4.3 Soundness with respect to model construction

Conversely, any complete open branch describes some model of its initial formulae. Such a soundness result follows from the invertibility of the logical rules, with respect to their logical nodes, established by the following lemma.

Lemma 4 *Let $\mathcal{M} = \langle T, \sigma\rangle$ be a temporal interpretation and $\mathcal{I}$ a temporal mapping with range T. For every logical rule*

$$\frac{\begin{array}{c}[s,t]\,F\\ s \leq t\end{array}}{\mathcal{B}_1 \qquad \mathcal{B}_2} \quad or \quad \frac{\begin{array}{c}[s,t]\,F\\ s \leq t\end{array}}{\mathcal{B}_1}$$

if $\langle \mathcal{M}, \mathcal{I}\rangle \models \mathcal{B}_i$ (for some $i = 1, 2$), then $\langle \mathcal{M}, \mathcal{I}\rangle \models [s,t]F$.

Proof. If $\mathcal{I}^*(t) < \mathcal{I}^*(s)$ then $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]F$ for any F. Hence we assume $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$. The different rules are treated as follows.

1. If F is $A \wedge B$, there is a single expansion and the result is straightforward.

2. If F is $A \vee B$, then the rule has two expansions. Let us assume that $\langle \mathcal{M}, \mathcal{I} \rangle$ satisfies the leftmost one: $[s,a]A$, $[s,a]B \vee A$ and $s \leq a \leq t$, where a is a fresh constant. Then obviously $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,a]A \vee B$ and $\langle \mathcal{M}, \mathcal{I} \rangle \models [a{+}1,t]A{\vee}B$, hence (whichever the position of $\mathcal{I}(a)$ with respect to $\mathcal{I}^*(s)$ and $\mathcal{I}^*(t)$) $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]A \vee B$. The other case is similar.

3. If F is $\Box A$, then $\langle \mathcal{M}, \mathcal{I} \rangle \models [s{+}1, finish]A$. Then clearly $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]\Box A$ (also in the case where $s = finish$).

4. If F is $\Diamond A$, and $\langle \mathcal{M}, \mathcal{I} \rangle \models [a]A$ for some a such that $\mathcal{I}(a) > \mathcal{I}^*(t)$, then $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]\Diamond A$.

5. If F is $A\mathcal{U}B$, then $\langle \mathcal{M}, \mathcal{I} \rangle$ satisfies $[a]B$, $[t+1, a-1]A$, $[s+1,t]A \vee B$, where a is a fresh constant such that $\mathcal{I}^*(t{+}1) \leq \mathcal{I}(a) = n$. Since $\langle \mathcal{M}, \mathcal{I} \rangle \models [t{+}1, a{-}1]A$, clearly $\mathcal{M}_{\mathcal{I}^*(t)} \models A\mathcal{U}B$. Let now $i \in [\mathcal{I}^*(s), \mathcal{I}^*(t{-}1)]$. Since for all $j \in [\mathcal{I}^*(s+1), \mathcal{I}^*(t)]$, $\mathcal{M}_j \models A \vee B$, also for all $j \in [i+1, \mathcal{I}^*(t)]$, $\mathcal{M}_j \models A \vee B$. We distinguish two cases:

   - If for all $j \in [i+1, \mathcal{I}^*(t)]$, $\mathcal{M}_j \models A$, then n itself is the "witness" for i: $n > j$, $\mathcal{M}_n \models B$ and for all j, $i < j < n$, $\mathcal{M}_j \models A$. Hence in this case $\mathcal{M}_i \models A\mathcal{U}B$.
   - Otherwise, let k be the smallest $j \in [i{+}1, \mathcal{I}^*(t)]$ such that $\mathcal{M}_k \not\models A$. Since $\mathcal{M}_k \models A \vee B$, $\mathcal{M}_k \models B$. Moreover $k > i$ and, by construction, for all j, $i < j < k$, $\mathcal{M}_j \models A$. Hence again $\mathcal{M}_i \models A\mathcal{U}B$.

6. The past time operators are treated similarly.

∎

Theorem 3 (Soundness w.r.t. model construction) *If $\mathcal{B}$ is a complete and open tableau branch and $\mathcal{I}$ is a solution of the set of the temporal constraints occurring in $\mathcal{B}$, then there is an interpretation $\mathcal{M}$ such that $\langle \mathcal{M}, \mathcal{I} \rangle \models \mathcal{B}$.*

Proof. The interpretation function σ is well defined, since, if $[s,t]\bot$ is in $\mathcal{B}$, then K contains $t \leq s-1$, because $\mathcal{B}$ is complete, so that there are no elements $i \in T$ with $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$.

By hypothesis, $\mathcal{I}$ satisfies all the temporal constraints in $\mathcal{B}$, so it rests to be shown that for any logical node $[s,t]F$ in $\mathcal{B}$, $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]F$. This is done by induction on the number of application of logical rules in $\mathcal{B}$. Note that if the constraint $t \leq s-1$ occurs in $\mathcal{B}$, then trivially $\langle \mathcal{M}, \mathcal{I} \rangle \models [s,t]F$. Hence, the only interesting cases arise when the constraint $s \leq t$ occurs in $\mathcal{B}$ (since $\mathcal{B}$ is complete, either $t \leq s-1$ or $s \leq t$ occurs in $\mathcal{B}$, for any $[s,t]F$ in $\mathcal{B}$). In the following, K denotes the set of constraints occurring in $\mathcal{B}$.

1. The base case is when there is no application of logical rules in $\mathcal{B}$, hence, since $\mathcal{B}$ is complete, the only logical nodes in $\mathcal{B}$ have the form $[s,t]F$, where F is a literal.

   - If F is an atom, the result holds by definition of σ.
   - If F is $\neg p$, assume that $\langle \mathcal{M}, \mathcal{I} \rangle \not\models [s,t]\neg p$, i.e. there exists $i \in T$ such that $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$ and $p \in \sigma(i)$. Hence, by definition of σ, there is a node $[s',t']p$ in $\mathcal{B}$ where $\mathcal{I}^*(s') \leq i \leq \mathcal{I}^*(t')$. Since $\mathcal{B}$ is complete, both $s \leq t$ and $s' \leq t'$ are in K (otherwise, $t \leq s-1$ or $t' \leq s'-1$ would be in K, contradicting the fact that neither $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$ nor $[\mathcal{I}^*(s'), \mathcal{I}^*(t')]$ are empty), and the conflict resolution rule is applied to $s \leq t$, $s' \leq t'$, $[s,t]\neg p$ and $[s',t']p$. Hence, either $t \leq s'-1$ or $t' \leq s-1$ is a node of $\mathcal{B}$. In both cases, the intersection of $[\mathcal{I}^*(s), \mathcal{I}^*(t)]$ and $[\mathcal{I}^*(s'), \mathcal{I}^*(t')]$ should be empty.

2. The inductive step follows from Lemma 4.

■

4.4 Complexity and the search for models of any finite size

Since the rules of the calculus are invertible, the search for a model can be done by construction of a single (canonical) tableau, following any strategy to choose the nodes to be expanded. It can be shown that the length of a branch in a tableau for $S \cup \{finish \leq start + k\}$ is polynomial in $n \times k$, where n is the size of S. Since testing whether a branch is open or not can be done in polynomial time [CLR90], a non deterministic algorithm implementing the proposed tableau system is $O(n \times k)$.[2] A deterministic algorithm is $O(2^{n \times k})$, since the number of branches must also be taken into account. However, the space required by depth first search is still polynomial in $n \times k$.

Suitable versions of Theorems 2 and 3 also hold when tableaux are initialized without any constraint of the form $finish \leq start + k$: if $\mathcal{M}$ is a model of S, then any canonical tableau for S has a finite open branch that is satisfiable in $\mathcal{M}$, and, conversely, any complete and open tableau branch is satisfiable. This gives a semi-decision procedure for testing satisfiability in models of finite size. In fact:

[2]This corresponds to the fact that, although the general plan-existence problem for STRIPS-like operators (see Section 5) is PSPACE-hard, planning is NP-complete when only polynomial length plans are considered.

1. Theorem 2 ensures that if F has a model $\mathcal{M}$, then there is a finite open branch (satisfied by $\mathcal{M}$). Hence, the search of finite open branches semidecides the satisfiability of F.

2. However, the complementary problem, i.e. the insatisfiability of F, is not semi-decided through tableau search. In fact, if F has no model, open branches of increasing depth may be constructed ad libitum.

However, the calculus also induces a decision procedure for checking LTL finite satisfiability, by means of a suitable initialisation of K. It is already known that finite satisfiability in LTL is decidable; for instance, this can be proved by use of Lemma 4.5 in [SC85]. This result can be adapted to the finite semantics: the following lemma is just a reformulation of Lemma 4.5 in [SC85].

Lemma 5 *Let F be an LTL formula, $T = \{t_0, ..., t_n\}$ and let $\mathcal{M} = \langle T, \sigma \rangle$ be a temporal interpretation. Suppose that t_i and t_j are two time points in T such that $i < j$ and, for any subformula A of F, $\mathcal{M}_{t_i} \models A$ iff $\mathcal{M}_{t_j} \models A$. Then the temporal interpretation $\mathcal{M}' = \langle T', \sigma' \rangle$, where T' is (a renaming of) $\{t_0, t_1, ..., t_{i-1}, t_j, ..., t_n\}$ and σ' is the restriction of σ to T' is such that:*

for any $t_x \in T'$, for any any subformula A of F, $\mathcal{M}'_{t_x} \models A$ iff $\mathcal{M}_{t_x} \models A$.

The above result has the following consequences:

1. For $\mathcal{M}$ and $\mathcal{M}'$ as in the lemma, F is true in $\mathcal{M}$ iff F is true in M'.

2. If F has model $\mathcal{M} = \langle T, \sigma \rangle$, then it has a model $\mathcal{M}^* = \langle T^*, \sigma^* \rangle$ where the cardinality of of T^* is not greater than the cardinality of the powerset of the set of subformulae of F, i.e. $2^{|sub(F)|}$.

Hence, if K is initialized to $\{finish \leq start + 2^{|sub(F)|}\}$, we immediately get a decision procedure for finite satisfiability in LTL.

This may be interesting from a theoretical standpoint, but clearly unpractical: if the bound k on the models size is exponential in n, a doubly exponential upper bound to the deterministic time complexity is obtained, and the space complexity is exponential in n.

5 Tabplan and planning algorithms

TabPlan is a prototype implementation of the above described tableau system, written in Standard ML. A modification of Bellman-Ford algorithm [CLR90] for testing satisfiability of sets of integer constraints is used, allowing for incremental tests, so that as soon as a new constraint is produced, the set is tested for consistency considering only the new local modification introduced. We do not have meaningful experimental results yet, showing how the system behaves with respect to formulae encoding planning problems, since this requires the study of refinements of the calculus suited for the special (syntactically restricted) form of planning problems encodings.

The initial tableau in TabPlan may have a more general form, in order to make the specification of the planning problem easier. A planning problem, in its general form, can be described by means of four sets of formulae:

- a set S of *initial state formulae*, describing the initial situation;
- a set F of *goal formulae*, describing what is expected hold in the final situation;
- possibly, a set T of *task* formulae, describing intermediate tasks that have to be accomplished before reaching the goal. In some problems, in fact, the activity is more significant than the goal and the specification of the goal state can even be equal to the initial state one, but intermediate tasks must be performed: "do this and that, then go back home".[3]
- a set G of *global assumptions*, describing what holds in every situation (description of action preconditions and effects, general truths, ecc.)

The input to TabPlan is therefore constituted by four finite sets of formulae in nnf, S, F, $T = \{A_1, ..., A_n\}$ and G, and $K = \{finish \leq start + k\}$, for some integer $k \geq 0$ corresponding to the maximal length of a suitable plan. An initial tableau for $\langle S, F, T, G, K\rangle$ is a one-branch tableau whose nodes are labelled by

$$\{[start]A \mid A \in S\} \cup \{[finish]A \mid A \in F\} \cup$$

$$\{[a_i]A_i \mid A_i \in T\} \cup \{[start, finish]A \mid A \in G\} \cup K$$

where $a_1, .., a_n$ are new constants different from $start$ and $finish$.

In order to generalize the semantics accordingly, the following notions are introduced.

Let A be a formula and $\mathcal{M} = \langle T, \sigma\rangle$ a temporal interpretation. Then:

- A is finally true in $\mathcal{M}$ ($\mathcal{M} \models_{finish} A$) iff $\mathcal{M}_k \models A$, where $k = max\{i \in T\}$, i.e. A is satisfied in the last time point of T.
- A is satisfiable in $\mathcal{M}$ ($\mathcal{M} \models_{task} A$) iff $\mathcal{M}_i \models A$, for some $i \in T$.
- A is valid in $\mathcal{M}$ ($\mathcal{M} \models A$) iff for every $i \in T$, $M_i \models A$.

The above notions can be obviously reduced to the notion of truth:

$$\mathcal{M} \models_{finish} A \text{ iff } \mathcal{M} \models_0 (A \wedge \Box\bot) \vee (\Diamond(A \wedge \Box\bot)$$

$$\mathcal{M} \models_{task} A \text{ iff } \mathcal{M} \models_0 (A \vee \Diamond A)$$

[3] Obviously, such tasks could be represented by means of goals of the form "having this and that done", but this is an unnecessary complication.

$$\mathcal{M} \models A \text{ iff } \mathcal{M} \models_0 (A \wedge \Box A)$$

With such notions, satisfiability of tuples of the above form can be defined: if $\Gamma = \langle S, F, T, G, K \rangle$, where S, F, $T = \{A_1, ..., A_n\}$ and G are finite sets of formulae and K is a set of temporal constraints, then $\langle \mathcal{M}, \mathcal{I} \rangle$ satisfies Γ ($\langle \mathcal{M}, \mathcal{I} \rangle \models \Gamma$) iff it satisfies the set

$$\{[start]A \mid A \in S\} \cup \{[finish]A \mid A \in F\} \cup$$

$$\{[a_i]A_i \mid A_i \in T\} \cup \{[start, finish]A \mid A \in G\} \cup K$$

Γ is satisfiable in $\mathcal{M}$ iff there exists a temporal mapping $\mathcal{I}$, for $\{a_1, ..., a_n, start, finish\}$, such that $\langle \mathcal{M}, \mathcal{I} \rangle$ satisfies the set above.

With these generalisation, all the results in Section 4 still hold.

TabPlan has specific rules for the treatement of the $\bigcirc, \odot, \overleftarrow{\bigcirc}, \overleftarrow{\odot}$ operators, i.e.:

$\dfrac{\begin{array}{c}[s,t]\bigcirc A\\ s \leq t\end{array}}{\begin{array}{c}t = finish\\ [s+1, finish]\,A\end{array} \qquad \begin{array}{c}t \leq finish - 1\\ [s+1, t+1]\,A\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\odot A\\ s \leq t\end{array}}{\begin{array}{c}t \leq finish - 1\\ [s+1, t+1]\,A\end{array}}$

and the symmetric ones for $\overleftarrow{\bigcirc}$ ane $\overleftarrow{\odot}$. TabPlan also implements some (obviously sound) shortcuts when expanding nodes with labels of the form $[a]$. E.g., the following logical rules are added to the calculus:

$\dfrac{[a]\,A \vee B}{[a]\,A \qquad [a]\,B}$	$\dfrac{[a]\,A\mathcal{S}B}{\begin{array}{c}[b]\,B\\ b \leq a-1\\ [b+1, a-1]\,A\\ b \in C\ fresh\end{array}}$	$\dfrac{[a]\,A\mathcal{U}B}{\begin{array}{c}[b]\,B\\ a+1 \leq b\\ [a+1, b-1]\,A\\ b \in C\ fresh\end{array}}$

In the rest of this section we give a brief overview of the main features of some important search based planning system, in order to illustrate the techniques that TabPlan has mutuated from them and some of the limits it allows to overcome.

Most of the planning languages in the non logical approach, whose semantics has been been given a formal characterisation, are extensions of the language of STRIPS, the first major planning system [FN71]. In such languages, the description of an action a consists of the specification of the preconditions for its executability, a set of formulae describing, for some relations R, the set of elements that will newly enjoy of R in the situation that results from the execution of a (the "add list") and formulae describing the set of elements that are going to lose some property R after executing a (the "delete list"). (In our formalism, the actions description makes part of the global assumptions.)

Syntactic restrictions are imposed on the formulae that are allowed to occur in the precondition, add and delete list, although different languages may vary in their expressive power.

Planners are planning algorithms that use the representation of the problem, given by action descriptions together with a description of the initial situation (corresponding to the initial state formulae) and goals (corresponding to goal formulae), in order to synthesise a plan, i.e. a sequence of actions that, if executed, would lead from the initial situation to the desired goals.[4]

The search of the plan can be performed in different ways. Mainly, the following features characterise different strategies:

- The search can either proceed in a data driven manner, starting from the initial situation (progression planners) or backward, taking the goals as the starting point (regression planners), considering those actions that achieve the goal and, in turn, their preconditions, until the initial situation is reached.

- The search space can be constituted either by the set of situations themselves, actions trasforming a situation into another one, and a plan is therefore a path leading from the initial situation to a situation where the goals are satisfied (linear planners), or by a set of *partial plans* (partial order planning). A partial plan is essentially a set of actions related by a (partial) ordering relation. Partial order planning starts with the empty plan and successively refines it, either by addition of new actions or new temporal constraints between them. The result of the search (the final plan) is itself a node in the search space, fulfilling some requirement that, roughly, ensures that every goal has been reached. This approach reflects the *least commitment principle*: postpone constraining decisions till they are effectively needed.

Although the preferability of one or the other of such approaches may strongly depend on the problem structure, consensus suggests that regression and partial order planning are generally more adequate to handle real world problems.

In [SPW92], a partial order planning algorithm (UCPOP), working by regression, is described. Succinctly, UCPOP starts with a plan consisting of two "dummy" actions: *start*, whose effects are the initial conditions, and *goal*, whose preconditions are the goals of the problem. Then, new actions and constraints are successively added to the plan, until all preconditions are satisfied. At each step, an "open" precondition P of an action a_i is chosen (that is not satisfied

[4]Note that intermediate tasks, corresponding to what have been called here task formulae, cannot be directly modeled in STRIPS-like languages, which lack the capability to refer to what happens between the initial state and the goal. They are taken into account by the *Hierarchical Task Networks* approach (see, for example [EHN94, Yan90]), as well as in [BK96b].

yet) and either a new action or an existing action, having such precondition as effect, is nondeterministically chosen and, if new, added to the plan. The plan is also added a *causal link* from the chosen action a_j to a_i, labelled by P, in order to record the fact that a_j is present in the plan exactly to produce P. After that, UCPOP resolves possible *threats*: there may be actions a_{threat} in the plan whose effect would destroy P. The threat is then solved either by *promotion*, adding the time constraint $a_{threat} < a_j$, or *demotion*, by addition of the constraint $a_i < a_{threat}$.

From the above sketchy description, it is apparent that a logical reading of UCPOP can be given, identifying its task as a sort of model construction. The correspondence of some of UCPOP principles and steps with TabPlan tableau method can then be be easily recognised. In particular, the allowed partial ordering of time points reflects a form of plan-space search planning, while the conflict resolution rules are usable also for solving threats. Moreover, the regression strategy followed by UCPOP can also be adopted by TabPlan. However, TabPlan is not a simple "logical rewriting" of UCPOP. In fact, different search strategies can be simulated by TabPlan, depending on the syntactic form of the encoding (more on this later on).

Furthermore, TabPlan has a richer expressive power than STRIPS-like formalisms:

- Intermediate tasks can be specified (by means of task formulae), therefore capturing within the same formalism planning problems where the activities to be performed are more important than the goal to achieve. TabPlan can also specify that some intermediate task are to be accomplished in a given order, by means, for example, of task formulae of the form $task_1 \wedge (\Diamond(task_2 \wedge \Diamond task_3))$, specifying that $task_1, task_2, task_3$ must be performed in that order. Similarly, TabPlan can specify that a given task must be accomplished at least a given number of times (by means of $task \wedge (\Diamond task \wedge \ldots (task \wedge \Diamond task) \ldots))$.

- From the standpoint of the search mechanism, TabPlan is in one respect "least committed" than UCPOP algorithm. When an "open" subgoal is chosen to be fulfilled, no choice is to be made between adding a new action or using an existing one: a new *state* is added to the plan in any case, but states do not bear a one-to-one correspondence with actions. No commitment is made on whether such new situation will eventually collapse into another one, where the same action is performed.

- As a consequence of the fact that a plan consists of a partially ordered set of *actions* – i.e. an action is identified with the state where it is performed – STRIPS-like formalisms cannot cope in a natural manner with problems requiring two or more actions to be performed contemporarily in order to achieve a desired effect.

In TabPlan, situations are not identified with actions (which are represented by means of ordinarily propositions), so that nothing prevents a model from containing a situation where two actions are performed. For example, if in order to open a door we must contemporarily push and hold its lock open, and the effect of such actions lasts until we release the door, this can be stated in TabPlan (regression) formalism by:

$$[start, finish]open\_door \rightarrow \\ (open\_door \wedge \neg release\_door)\mathcal{Z}(do\_push \wedge do\_hold\_lock)$$

Furthermore, parallel actions can be modeled.

- While regression planning is expressible in Tabplan, Tabplan does not force it. Data driven plan construction may be as well expressed. In fact, the syntactic form the problem specification is given strongly influences the search strategy. This can be better illustrated by an example. Assume that, among the actions the agent can perform, there are putting the right shoe on (put_rshoe_on), taking the right shoe off ($take\_rshoe\_off$), putting the right sock on (put_rsock_on) and taking the right sock off ($take\_rsock\_off$). The effects of wearing the right shoe and the right sock are represented by the atoms $rshoe$ and $rsock$, respectively. Whichever the search strategy to be used, action preconditions are expressed as follows:

$$\begin{array}{l} [start, finish]\, put\_rshoe\_on \rightarrow \neg rshoe \wedge rsock \\ [start, finish]\, put\_rsock\_on \rightarrow \neg rsock \\ [start, finish]\, take\_rshoe\_off \rightarrow rshoe \\ [start, finish]\, take\_rsock\_off \rightarrow rsock \end{array}$$

As a consequence, an action is performed only if its preconditions are fulfilled.

If regression planning is preferred, then the formulation of the effect of an action is "effect centered", as follows:

$$\begin{array}{l} [start, finish]\, rsock \rightarrow (rsock \wedge \neg take\_rsock\_off)\, \mathcal{Z}\, put\_rsock\_on \\ [start, finish]\, rshoe \rightarrow (rshoe \wedge \neg take\_rshoe\_off)\, \mathcal{Z}\, put\_rshoe\_on \\ [start, finish]\, \neg rsock \rightarrow (\neg rsock \wedge \neg put\_rsock\_on)\, \mathcal{Z}\, take\_rsock\_off \\ [start, finish]\, \neg rshoe \rightarrow (\neg rshoe \wedge \neg put\_rshoe\_on)\, \mathcal{Z}\, take\_rshoe\_off \end{array}$$

The resulting specification contains only past time operators.

By contrast, in order to make TabPlan simulate progression planning, the description will be "action centered", corresponding to usual "frame and effect axioms" and making use of future time operators only:

$$
\begin{aligned}
&[start, finish]\, put\_rsock\_on \vee (rsock \wedge \neg take\_rsock\_off) \rightarrow \\
&\hspace{6cm} (rsock\, \mathcal{W}\, take\_rsock\_off) \\
&[start, finish]\, put\_rshoe\_on \vee (rshoe \wedge \neg take\_rshoe\_off) \rightarrow \\
&\hspace{6cm} (rshoe\, \mathcal{W}\, take\_rshoe\_off) \\
&[start, finish]\, take\_rsock\_off \vee (\neg rsock \wedge \neg put\_rsock\_on) \rightarrow \\
&\hspace{6cm} (\neg rsock\, \mathcal{W}\, put\_rsock\_on) \\
&[start, finish]\, take\_rshoe\_off \vee (\neg rshoe \wedge \neg put\_rshoe\_on) \rightarrow \\
&\hspace{6cm} (\neg rshoe\, \mathcal{U}\, put\_rshoe\_on)
\end{aligned}
$$

6 Conclusions and related work

In this work, the possibility of using linear temporal logic as a planning language is investigated, in the view that a plan is a finite model of the specification of the problem. Whith respect to the formalisms used in the search based approach to planning, the use of LTL shares the advantages of any logical approach: a formal semantics, generality and expressivity. For example, the same language can be used to specify standard planning problems as well as problems of with intermediate tasks to be accomplished; no committment is made about the possibility of performing more actions contemporarily or in parallel, nor on the syntactic form of formulae describing actions, initial conditions and goals; the specification of a given problem can assume mainly two syntactically different forms, each of them corresponding to, and causing the execution to adopt one of the two main search strategies, goal driven and data driven. Furthermore, using a well studied logic allows the exploitment of meta-theoretical results. In this case, we know that completeness (with respect to plan construction of any length) is not lost if the search is restricted to consider plans whose maximal length is $2^{|sub(S)|}$, where S is the problem specification.

On the other hand, the tableau calculus by means of which model search is performed allows the embedding of different strategies and techniques mutuated from the search based approach to planning, depending on the encoding of the problem. In particular, plan space search can be simulated, since the calculus uses labelled formulae, labels corresponding to time intervals where the corresponding formulae are true, and time points are only partially ordered.

Different proof systems for linear temporal logic can be found in the literature, that can be turned into model search methods (the underlying model of time being however always infinite). Most of them (for example, [Wol85, BCM$^+$92, BFG$^+$89]) are essentially based on the following equivalences

$$\Diamond A = A \vee \bigcirc \Diamond A \qquad A \mathcal{U} B = B \vee (A \wedge \bigcirc (A \mathcal{U} B))$$

where $\bigcirc$ is the "next" operator and $\Diamond$ and $\mathcal{U}$ have the weak semantics, including the present time point. Rewriting a formula of the form $\Diamond A$ as $A \vee \bigcirc \Diamond A$ leads to choose whether "executing" A in the present time point or postponing its execution, the same problem passing on to the next time point. Consequently, in the model description that results from the construction, the ordering of time points is always total. Moreover, such systems must be equipped with some mechanism to check whether all "eventualities" (formulae in the scope of a $\Diamond$ or $\overleftarrow{\Diamond}$, or in the right scope of a $\mathcal{U}$ or $\mathcal{S}$ operator) are sooner or later satisfied.

For example, the executable temporal language METATEM [BFG+89] is based on the paradigm "declarative past implies imperative future". A formula of the form $Past \rightarrow Future$ is executed by checking the truth of $Past$ in the present history (the model built so far) and, if it is true, the model is extended in order to make $Future$ true. Its application to planning and scheduling has been considered [BFGH91] but the "declarative past implies imperative future" reading of formulae and their execution techniques correspond to a form of total-order, forward-chaining planning.

The systems reported in [SGL97, SGL], carrying on the work in [HI94], are significantly different from the above ones. As illustrated below, the present paper integrates several important points of such works. A tableau branch, in [SGL97, SGL], consists of a set of nodes of the form $[s,t]C$, where s and t denote time points and C is a *set* of formulae (disjunctively interpreted), and a set of constraints on time points defining a partial order. In [SGL97], where only unary temporal operators are conidered, the construction of tableau branches terminates, and a set of "resolution rules" can be applied to nodes consisting of classical clauses,[5] causing the addition of new constraints on time points. Branch closure is reduced to the satisfiability of the set of integer constraints in the branch. The introduction of the "since" and "until" operators – that are however essential in coding planning problems – raises new difficulties. In [SGL] the tableau expansion rules for such operators are mainly based on the equivalences $A\mathcal{U}B = B \vee (A \wedge \bigcirc(A\mathcal{U}B))$ and the symmetric one for $\mathcal{S}$, but an ingenious rewriting of the "contracted" formula into a propositional letter guarantees termination. After the terminating tableau expansion, branch closure is checked by reduction to a model checking problem in CTL with fairness constraints, in the style of [BCM+92]. Because of the special form of the set of active formulae in the branch, the model checking problem is somewhat simplified. It would be interesting to investigate whether focusing on finite models could lead to simpler solutions. In this case, in fact, some hard problems disappear, such as the problem of fairness conditions (in connection with formulae of the form $\Box\Diamond A$).

This work lays the theoretical grounds for the use of TabPlan to solve planning problems. In order to have a practical system, one of the first issues to be addressed is the extension to a limited first order language. In fact, although from a theoretical standpoint propositional logic suffices to represent planning problems over finite domains, needless to say that treating an existential quanti-

[5] The conflict resolution rules of this paper are a simplification of such rules. Also the resolution of *threats* in UC-POP can be seen as a particular form of resolution rule.

fied formula like the finite disjunction of its instances is not an excellent solution. It would be a gross violation of the *least commitment* principle.

Furthermore, the class of formulae involved in different encodings of planning problems must be analysed, aiming at defining refinements of the calculus for such formulae and improving TabPlan performance. This would allow us to collect experimental results on significant planning problem examples.

Finally, the possibility to apply TabPlan to other domains, such as, for example, the management of dynamical integrity constraints in data bases, is to be investigated.

Acknowledgements. The authors are strongly indebted with Jean Goubault-Larrecq for reading a first version of this work and making helpful remarks.

References

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[BFG+89] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: a framework for programming in temporal logic. In *Proc. of REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *LNCS*. Springer, 1989.

[BFGH91] H. Barringer, M. Fisher, D. Gabbay, and A. Hunter. Meta-reasoning in executable temporal logic. In *Proc. of the Second Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1991.

[BK96a] F. Bacchus and F. Kabanza. Using temporal logic to control search in a forward chaining planner. In M. Ghallab and A. Milani, editors, *New Directions in AI Planning*, pages 141–156. IOS Press (Amsterdam), 1996.

[BK96b] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1215–1222. AAAI Press / The MIT Press, 1996.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT press, 1990.

[CO96] A. Cesta and A. Oddi. DDL.1: a formal description of a constraint representaton language for physical domains. In M. Ghallab and A. Milani, editors, *New Direction in AI Planning*, pages 341–352. IOS Press, 1996.

[CT91] K. Currie and A. Tate. O-Plan: the open planning architecture. *Journal of Artificial Intelligence*, 52:49–86, 1991.

[EHN94] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. In *Proceedings of AAAI-94*, 1994.

[FN71] R. E. Fikes and N.J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[FO95] M. Fisher and R. Owens. An introduction to executable modal and temporal logics. In M. Fisher and R. Owens, editors, *Executable modal and temporal logics (Proc. of the IJCAI'93 Workshop)*, volume 897 of *LNAI*, pages 1–20. Springer, 1995.

[HI94] R. Hähnle and O. Ibens. Improving temporal logic tableaux using integer constraints. In D. M. Gabbay and H. J. Ohlbach, editors, *Proceedings of the First International Conference on Temporal Logic (ICTL 94)*, volume 827 of *LNCS*, pages 535–539. Springer, 1994.

[KS92] 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.

[KT95] J. Koehler and R. Treinen. Constraint deduction in an interval-based temporal logic. In M. Fisher and R. Owens, editors, *Executable Modal and Temporal Logics, (Proc. of the IJCAI'93 Workshop)*, volume 897 of *LNAI*, pages 103–117. Springer, 1995.

[Mas93] M. Masseron. Generating plans in linear logic: II. A geometry of conjunctive actions. *Theoretical Computer Science*, 113:371–375, 1993.

[MTV93] M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic: I. Actions as proofs. *Theoretical Computer Science*, 113:349–370, 1993.

[Rei91] 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.

[SB96] B. Stephan and S. Biundo. Deduction based refinement planning. In B. Drabble, editor, *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 213–220. AAAI Press, 1996.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[SGL] P.H. Schmitt and J. Goubault-Larrecq. A tableau system for full linear temporal logic. Draft, available at: http://www.dyade.fr/fr/actions/vip/jgl/ltl2.ps.gz.

[SGL97] P.H. Schmitt and J. Goubault-Larrecq. A tableau system for linear-time temporal logic. In E. Brinksma, editor, *3rd Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97)*, LNCS. Springer Verlag, 1997.

[SPW92] J. Scott Penberthy and D.S Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. of the Third Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 103–114. Morgan Kauffman Publ., 1992.

[Wol85] P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28:119–152, 1985.

[Yan90] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6:12–24, 1990.