Bounded Model Search in Linear Temporal Logic and its Application to Planning

Serenella Cerrito[1] and Marta Cialdea Mayer[2]*

[1] Université de Paris-Sud, LRI, Bât 490,
F-91405 Orsay Cedex, France.
e-mail: serena@lri.fr

[2] Dipartimento di Informatica e Automazione, Università di Roma Tre,
via Vasca Navale 79, 00146 Roma, Italia.
e-mail: cialdea@dia.uniroma3.it

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.

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.

In [12] 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, 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 [16], linear logic [14, 15], temporal logics [21, 13]. Works using modal temporal logics rely on the branching model of time of interval-based temporal logics.

* This work has been partially supported by MURST, ASI (Agenzia Spaziale Italiana) and CNR (SCIxSIA Project).

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, [8, 19]). 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 4), 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 to lay the grounds for the design of a logical language and an associated calculus for plan construction having, on one side, the declarativity, expressiveness and flexibility typical of the logical approach, together with a clean semantics based on a natural model of time and action, and, on the other, the capability of embedding search-based techniques, typically partial order, regression and least commitment. Viewing a plan as a finite model of the problem specification, this work proposes the use of linear temporal logic (LTL) as the specification language (differently, [12] uses propositional classical logic). The main advantage of the use of LTL as a planning language derives from its rich expressive power[1] and the underlying simple model of time.

The search for finite LTL models is carried out by means of tableau construction. Semantical tableaux 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 the design of mixed-initiative systems can be a good choice in order to control the complexity of real world planning problems. Moreover, tableau methods have proved 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 [18]. However, while temporal structures are alway infinite in that work, we focus on finite temporal models, that are obviously more suitable to represent plans.

Planning is a notoriusly hard problem: plan existence is PSPACE-complete for propositional STRIPS-planning [5]. However, the existence of polynomial-length plans is NP-complete [12], hence practical solutions are often required not to exceed a given (polynomial) length known in advance. For this reason,

[1] Mainly, with respect to classical propositional logic, the "since and "until" operators add the strength of a sort of bounded quantification.

this work focuses on *bounded* planning problems, i.e., in the temporal logic approach, the search of finite models whose size is not greater than a given limit. Whenever such a limit is not known in advance, the shortest plan can be found by iterative deepening search. Moreover, from a theoretical standpoint, the calculus can also be used to check finite satisfiability *tout court*, just by fixing a suitable (exponential) limit to the plan length (see Section 3.4).

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.

2 The Tableau System for Linear Temporal Logic

The language of linear temporal logic we consider extends classical propositional logic, with the logical operators $\neg, \wedge, \vee$, $\perp$ (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;
- $\sigma : T \rightarrow \mathcal{P}(P)$ 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 at i.

The satisfiability relation $\mathcal{M}_i \models A$, for $i \in T$, is inductively defined by addition of the following clauses to the usual ones for the classical connectives:

1. $\mathcal{M}_i \models \Box A$ iff for all $j > i$, $\mathcal{M}_j \models A$.
2. $\mathcal{M}_i \models \Diamond A$ iff there exists $j \in T$ such that $j > i$ and $\mathcal{M}_j \models A$.
3. $\mathcal{M}_i \models A\mathcal{U}B$ iff $\exists j \in T$ such that $j > i$ and $\mathcal{M}_j \models B$ and for any k with $i < k < j$ $\mathcal{M}_k \models A$.
4. $\mathcal{M}_i \models \overleftarrow{\Box} A$ iff for all $j < i$, $\mathcal{M}_j \models A$.
5. $\mathcal{M}_i \models \overleftarrow{\Diamond} A$ iff there exists $j \in T$ such that $j < i$ and $\mathcal{M}_j \models A$.
6. $\mathcal{M}_i \models A\mathcal{S}B$ iff $\exists j \in T$ such that $j < i$ and $\mathcal{M}_j \models B$ and for any k with $j < k < i$ $\mathcal{M}_k \models A$.

Truth is satisfiability in the initial state: a formula A is true in $\mathcal{M}$ (and $\mathcal{M}$ is a model of A) iff $\mathcal{M}_0 \models A$. Truth of sets of formulae is defined as usual.

Note that, 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 definable. Although also the $\Diamond$ and $\overleftarrow{\Diamond}$ operators are definable, we prefer to give them a separate treatment in the tableau system. The semantics of such "existential" operators is again the strong one, so that, for example, $\Box\Diamond A$ can be true only in temporal structures consisting of a single time point (it is in fact always true in such cases).

A formula is in *negation normal form* (nnf) iff no logical operator is in the scope of a negation. Two formulae A and B are equivalent iff for all $\mathcal{M}$ and i, $\mathcal{M}_i \models A$ iff $\mathcal{M}_i \models B$. It can easily be shown that, under this strong notion of equivalence, every formula can be transformed into an equivalent formula in nnf, by applying the usual equivalences for $\rightarrow, \wedge, \vee, \Box, \overleftarrow{\Box}, \Diamond, \overleftarrow{\Diamond}$, and:

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

In the rest of this section a tableau system is defined, that allows one to test whether a set LTL formulae in nnf admits models whose underlying temporal structure does not exceed a given size. In case of a positive answer, a model can be easily extracted from the tableau construction. The restriction to nnf formulae is introduced only to simplify the presentation of the rules.

Let $C = \{start, finish, d_1, d_2, d_3, ...\}$ be a set of *constants* (intuitively denoting time points). A *state* is any expression of the form $c+n$, for $c \in C$ and $n \in Z$. The set of states is denoted by Σ. It is intended that $C \subset \Sigma$ (c can be rewritten as $c+0$). If $s, t \in \Sigma$, then $s \leq t$ is a *temporal constraint*. A *labelled formula* is an expression of the form $[s,t]A$, where $s, t \in \Sigma$ and A is an LTL formula in nnf. $[s,s]A$ will be abbreviated by $[s]A$.

Tableau nodes are labelled either by temporal constraints or labelled formulae (that are called *logical nodes*). If S is a finite set of formulae in nnf and $K = \{finish \leq start + k\}$ for some integer $k \geq 0$ (representing the maximal size of the searched models), then tableaux for $S \cup K$ are initialized with the set $\{[start]A \mid A \in S\} \cup K$ and expanded by application of the rules in Table 1, where c denotes an element of C, and $s, t, s', t', ..$ elements of Σ. The set of nodes occurring above the line of a rule is called the *premise* of the rule, while the sets of nodes occurring below are the *expansions* of the premise.

Note that a sort of contraction is implicit in the β-rule: 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. The intuition behind the β-rule is the following: either A is true in the whole interval (leftmost branch), or there exists a smallest time point c in the interval where A is false, hence B is true; since c is chosen to be the first of such points, A is true in the (possibly empty) subinterval before it. The rule could also be formulated in a symmetric way, distinguishing two cases according to whether A is true at s or B is true at s. However, this would force the rule to be reapplied once for each point of the interval; on the contrary, with the proposed asymmetric formulation the interval may be cut into larger pieces at each application (obviously, the behaviour is the same in the worst case, where A and B are interleaved). The β-rule is a delicate point: note that it is indirectly charged to expand also $\mathcal{U}$- and $\mathcal{S}$-formulae.

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

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

Table 1. Tableau expansion rules

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 distinguishes the cases where an interval is empty or not. Its role is to provide the preconditions for the application of the logical and resolution rules. Intuitively, it is useless – and sometimes even incorrect – to expand a node $[s,t]A$ when the interval is empty and, given two nodes $[s,t]p$, $[s',t']\neg p$, there is no conflict to be solved if either $[s,t]$ or $[s',t']$ (or both) are empty. Note that such rule could be dispensed with, and a corresponding branching added to most of the other rules, handling the case where the considered interval is empty. Its distinction as a separate rule makes the formulation of the calculus more compact and clearer: obviously, a test on the "emptyness" of an interval $[s,t]$ needs to be done just once in a branch, independently on the number of logical nodes labelled by $[s,t]$.

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 $\mathcal{I}^*(c+n) = \mathcal{I}(c) + n$ for every $c \in C, n \in Z$.*

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 $min\{\mathcal{I}(c) \mid c \in C\} = \mathcal{I}(start) = 0$ and $max\{\mathcal{I}(c) \mid c \in C\} = \mathcal{I}(finish) = k$. 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.

Later on (Lemma 2) we show that every (non redundant) infinite branch is closed. Hence, we are only concerned with checking closure for finite branches, that amounts to checking satisfiability of a finite set of integer constraints. This can be done by means of well known graph algoritms (see any standard textbook on algorithms, e.g. [7]).

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.*

Here follows an example, showing the partial development of a tableau for $A = \Diamond p \wedge \Box(\neg p \vee q)$. Below, the application of the interval rule to intervals of the form $[c]$ is not shown, and neither are obvious premises of the form $c \leq c$. The nodes are numbered in order to comment the tableau. We assume that the tableau is initialized with some limit k, for any $k \geq 1$.

$$1.\ \mathit{finish} \leq \mathit{start} + k$$
$$2.\ [\mathit{start}]\, A$$
$$3.\ [\mathit{start}]\, \Diamond p$$
$$4.\ [\mathit{start}]\, \Box(\neg p \vee q)$$
$$5.\ [d_1]\, p$$
$$6.\ \mathit{start} + 1 \leq d_1$$
$$7.\ [\mathit{start}+1, \mathit{finish}]\, \neg p \vee q$$

8. $\mathit{finish} \leq \mathit{start}$		9. $\mathit{start} + 1 \leq \mathit{finish}$
	10. $[\mathit{start}+1, \mathit{finish}]\, \neg p$	13. $\mathit{start} + 1 \leq d_2$
11. $d_1 \leq \mathit{start}$	12. $\mathit{finish} \leq d_1 - 1$	14. $d_2 \leq \mathit{finish}$
		15. $[\mathit{start}+1, d_2 - 1]\, \neg p$
		16. $[d_2]\, q$
		17. $[d_2 + 1, \mathit{finish}]\, \neg p \vee q$
	18. $d_2 - 1 \leq \mathit{start}$	19. $\mathit{start} + 1 \leq d_2 - 1$
20. $\mathit{finish} \leq d_2$	21. $d_2 + 1 \leq \mathit{finish}$	

Nodes 3 and 4 are the expansion of 2, and their expansions are 5, 6 and 7, respectively. Nodes 8 and 9 result from the application of the interval rule to 7; the branch with node 8 is closed, since, implicitely, $d_1 \leq \mathit{finish}$. Nodes 10 and 13–17 are the branches obtained from 7 and 9. The conflict resolution rule is applied to 5, 9 and 10, giving 11 and 12, that both close. Nodes 18–19 and 21–21 derive from 15 and 17, respectively, by the interval rule. The branches passing through 19 and 21 are not further developed. The open and complete branch ending at 20 represents the smallest (partial) model, with points $0 = \mathcal{I}(\mathit{start}) < 1 = \mathcal{I}(d_1) = \mathcal{I}(d_2) = \mathcal{I}(\mathit{finish})$, with both p and q holding at 1. One of the further expansions of the branch passing through 21 would yeld a more general description, with $0 = \mathcal{I}(\mathit{start}) < 1 = \mathcal{I}(d_2) \leq \mathcal{I}(d_1)$, where p holds at $\mathcal{I}(d_1)$ and q holds in the whole interval $[1, \mathcal{I}(\mathit{finish})]$. The development of 19 explores models where $d_1, d_2 > 1$.

3 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. Complete proofs can be found in [6].

3.1 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. As already remarked, the existence of a solution for a given set of constraints can be tested by means of shortest-path graph algorithms.

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 $X_0, X_1, ...$ 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 $X_0, X_1, ..., X_{k+1}$, 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 $0 \leq I^*(s) \leq I^*(t) \leq \mathcal{I}(finish)$.*

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 open branch is finite.*

Proof. We give here only a proof of the first item. 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.

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

3.2 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 extension $\mathcal{I}'$ 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 $\mathcal{I}$). Here we show the treatment of the β-rule and the $\mathcal{S}$-rule.

1. 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 $\langle \mathcal{M}, \mathcal{I}' \rangle \models [a+1, t]A \vee B$.
2. 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.

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

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 $\mathcal{M}$.*

3.3 Soundness with respect to Model Construction

Conversely, any complete open branch describes some model of its initial formula. 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 $\mathcal{I}^*(s) \leq \mathcal{I}^*(t)$. Here, we prove the lemma only for the case of the $\mathcal{U}$-rule.

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 $\mathcal{M}_i \models A\mathcal{U}B$.

Theorem 3. *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. First of all, observe that by Lemma 1 the branch $\mathcal{B}$ is finite. Let T be the (finite) range of $\mathcal{I}$ and $\mathcal{M} = \langle T, \sigma \rangle$, where σ is such that:

> for each atom p and $i \in T$, $p \in \sigma(i)$ iff there is a node $[s,t]p$ in $\mathcal{B}$ such that $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$.

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.

3.4 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, 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.

However, the calculus also induces a decision procedure for checking LTL finite satisfiability. It is already known that finite satisfiability in LTL is decidable; for instance, this can be proved by use of Lemma 4.5 in [20] (that can be easily adapted to our finite semantics). Roughly, the lemma says that, given any model $\mathcal{M}$ of an LTL formula F, if two time points satisfy exactly the same subformulae of F then they may collapse (and the interval between them disappear), so that a smaller model $\mathcal{M}'$ may be obtained. As a consequence of such a result, if F has a model, then it has a model whose size does not exceed the cardinality of the powerset of the set of its subformulae, i.e. $2^{|sub(F)|}$. In our framework, this implies that by setting $K = \{finish \leq start + 2^{|sub(F)|}\}$ we immediately get a decision procedure for finite satisfiability in LTL.

[2] This corresponds to the fact that, although the general plan-existence problem for STRIPS-like operators (see Section 4) is PSPACE-complete, planning is NP-complete when only polynomial length plans are considered.

This may be interesting from a theoretical standpoint, but clearly unpractical: if the bound k on the model 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.

4 Planning in LTL

In this section, we give a brief overview of the main features of some important search based planning system, in order to illustrate the advantages induced by the use of LTL as a specification language for planning problems and the use of the proposed tableau calculus as the basis for a plan search engine.

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 [10]. 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"). 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 and goals, 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. The search for 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 transforming 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 actually 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 [19], 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 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 $a_i < a_{threat}$.

The use of LTL as a specification language allows one to encode planning problems in a very flexible way. In general, the encoding of a planning problem in LTL consists of the following sets of formulae:

- a set S of *initial state formulae*, describing the initial situation, where no temporal operator occurs;
- a set F of *goal formulae*, describing what is expected hold in the final situation, each of which having the form $\Diamond^-(A \wedge \Box\bot)$, where $\Diamond^- B \equiv_{def} (B \vee \Diamond B)$;
- a set G of *global assumptions*, describing what holds in every situation (description of action preconditions and effects, general truths, etc.), having the form $A \wedge \Box A$.

Moreover, a specification can include a set T of *task* formulae, describing intermediate tasks that have to be accomplished before reaching the goal, in the form $\Diamond^- A$. 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". Intermediate tasks 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 [9, 23]), as well as in [1].[3]

The richer expressive power of LTL as a planning language, with respect to STRIPS-like formalisms, can be exploited in several other directions. For example, 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 LTL, on the contrary, situations are not identified with actions (which are represented by means of ordinary propositions), so that nothing prevents a model from containing a situation where two or more actions are performed. Furthermore, parallel actions can be modeled.

[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.

The same advantages are shared by the "planning as satisfiability" paradigm, proposed by Kautz and Selman [12], the logical language used to encode planning problems being propositional logic. However, with respect to the use of propositional logic, in an LTL specification time is implicitly represented and the presence of the binary temporal operators "since" and "until" gives the language a very rich expressive power, resulting in the possibility to assert what must hold in a whole interval.

When model search for LTL formulae is performed by means of the proposed tableau system, different plan search strategies may be simulated, by use of different ways of writing global assumptions. For instance, the regression strategy followed by UCPOP may be simulated by meand of "regression encodings", where effect and frame axioms have the form $\Box(p \rightarrow (\neg B \mathcal{S} A) \vee \overleftarrow{\Box}(p \wedge \neg B))$, where A (resp. B) encodes all the conditions that may lead p to become true (resp. false): if p is true sometime, then it must be the case that either some action was performed before (A), having p as effect, and such an effect has not been destroyed (by B) since then, or p has always been true and no action destroyng p has ever been performed. The search in the tableau resulting from the expansion of such regression axioms is goal driven. The partial ordering of time points in tableau construction reflects a form of plan-space search planning, while the conflict resolution rules are also used for solving UCPOP "threats". Data driven plan construction may be expressed as well, by means of global assumptions ("progression encodings") roughly stating that if an action is performed at point i, then its effects hold at point $i+1$ (or even in a whole interval, until (possibly) such effects are destroyed), and classical frame conditions for all literals. A detailed descritption of such encodings, and the induced plan search mechanisms, would override space limits and will be given elsewhere.

The application of LTL to planning has been considered in other works. For example, [3] applies the executable temporal language METATEM [2] to planning and scheduling. F. Bacchus and F. Kabanza [1] use a version of temporal logic to specify temporally extended goals as sets of acceptable sequences of states, i.e. temporal models, and define correct plans as those whose execution results in one of such sequences.[4] In both cited works, only a form of linear, data driven planning is obtained.

5 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. With respect to 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. Furthermore, using a well studied logic

[4] Temporal logic is not used to encode the planning problem entirely: actions are described in an ADL format, and an *Expand* operation is used, that, applied to a state s, generates all the successors of s that are produced by performing any allowed action.

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, [22, 4, 2]) 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.

The systems reported in [18, 17], carrying on the work in [11], are significantly different from the above ones. Several important points of such works are resumed in the present paper. In [18], where only unary temporal operators are considered, the construction of tableau branches terminates and 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 is however essential in coding planning problems – raises new difficulties. In [17] 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 [4]. Because of the special form of the set of active formulae in the branch, the model checking problem is somewhat simplified. Surely, 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 LTL, with the proposed tableau system, to solve planning problems. A prototype implementation exists, TabPlan. It is written in Standard ML and uses a modification of Bellman-Ford shortest path algorithm to check branch closure, allowing for incremental tests, so that as soon as a new constraint is produced, the set is tested for consistency considering only what has been newly modified. Moreover, some obvious short-cuts are added to the rule set. In order to have a practical planning system and collect experimental results on significant planning problem examples, however, some theoretical work still has to be done.

One of the first issues to be addressed is the characterization of the class of formulae involved in different encodings of planning problems. In fact, one does not expect that a general purpose system for full temporal logic over finite domains can compete with special purpose planners. One of the main reasons is that the encoding of a planning problem never exploits the full expressive power of LTL. For example, the nesting of temporal operators is always limited; regression encodings mainly use past time operators and progression ones future time operators; the occurrences of β-subformulae can be recognized to have a regular structure, that allows one to solve the general problem raised by β-chains. Hence, refinements of the calculus for planning formulae can be defined, in order to improve TabPlan performance.

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

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 paper and making helpful remarks. We also thank Wolfgang Gehrke, for implementing TabPlan first prototype.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. T. Bylander. Complexity results for planning. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 274–279, 1991.

6. S. Cerrito and M. Cialdea Mayer. TabPlan: Planning in linear temporal logic. Technical Report 1141, Université de Paris Sud, Laboratoire de Recherche en Informatique, 1997.
7. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT press, 1990.
8. K. Currie and A. Tate. O-Plan: the open planning architecture. *Journal of Artificial Intelligence*, 52:49–86, 1991.
9. K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. In *Proceedings of AAAI-94*, 1994.
10. 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.
11. 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.
12. 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.
13. 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.
14. M. Masseron. Generating plans in linear logic: II. A geometry of conjunctive actions. *Theoretical Computer Science*, 113:371–375, 1993.
15. M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic: I. Actions as proofs. *Theoretical Computer Science*, 113:349–370, 1993.
16. 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.
17. P.H. Schmitt and J. Goubault-Larrecq. A tableau system for full linear temporal logic. Draft.
18. 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.
19. 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.
20. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
21. 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.
22. P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28:119–152, 1985.
23. Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6:12–24, 1990.