ROMA TRE
Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy.

Planning as model construction in linear temporal logic

Marta Cialdea Mayer[1], Serenella Cerrito[2], Amedeo Cesta[3]

RT-DIA-23-1997 **May 1997**

1. Dipartimento di Informatica e Automazione,
Università di Roma Tre,
Via della Vasca Navale, 84
00146 Roma, Italy
e-mail: cialdea@inf.uniroma3.it

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

3. IP-CNR, National Research Council of Italy,
Viale Marx 15, I-00137 Rome, Italy.
e-mail: amedeo@pscs2.irmkant.rm.cnr.it

Marta Cialdea's work is partially supported by the Italian MURST project "Rappresentazione della Conoscenza e Meccanismi di Ragionamento" and by ASI – Agenzia Spaziale Italiana. Amedeo Cesta's work is partially supported by ASI – Agenzia Spaziale Italiana, CNR Committee 12 on Information Technology (Project SARI), and CNR Committee 04 on Biology and Medicine.

ABSTRACT

In this work we propose a logical approach to planning, based on the view of a planning activity as the search for a finite model of the specification of the planning problem. Planning problems are represented in a modal temporal language, based on a discrete, linear model of time, whose execution mechanism exploits model construction techniques related to analytic tableaux. The tableau system is based on the use of prefixes that, intuitively, label each formula as being true in a particular state. This approach allows the exploitation of methods and results from search-based approaches, such as plan-space search, regression and least commitment. A partial ordering of time points is in fact allowed in the calculus, by means of a set of temporal constraints that is built up step by step as the construction of a tableau branch proceeds. A form of soundness and completeness with respect to finite model construction are established, constituting the grounds for a decision procedure that is parametric w.r.t. the maximal time length a plan is allowed to reach.

1 Introduction

In recent literature, increasing attention has been devoted to the problem of the expressive power of planning languages. This is motivated by the aim of addressing more ambitious tasks by endowing planners with the ability to naturally model a variety of domains. Some attempts have been done to use formalisms different from classical STRIPS-like ones. The Task-Formalism of O-Plan [5], the logical language TPL [12], the domain description language DDL.1 [4] are different examples of such languages. These languages are able to represent time explicitly, model the behaviour of physical objects, represent resources, etc. As far as the generation of solutions is concerned, both O-Plan and the planner associated with DDL.1 refer to a search-based and formalism-tailored approach, while TPL is based on theorem proving. A careful analysis of these works shows how differences between logic-based and search-based approaches are not that conflicting but can synergically interact.

Generally speaking, a desiderata for a modern planning system could be the integration of the expressive power and flexibility of logical approaches with the efficiency of search-based methods. This is the rationale that characterises the present work, that, although situated in the logical side, devotes attention to the introduction of hooks and hints allowing the integration of results from classical planning systems research.

Usually, the logical approach to planning is based on deduction, and plan generation is carried out by constructively *proving* plan specification formulae. This feature is shared by most works based on modal temporal logics, mainly interval-based temporal logic [12, 10]. The present work proposes a different logical approach to planning, based on discrete, linear temporal logic. Mainly, similarly to [9], we consider that a planning activity may be seen as the search for a finite model of the specification of the planning problem. In other words, planning amounts to prove that $preconditions \wedge goals$ has a finite temporal model and such a model represents, in fact, a plan achieving the desired goals. This view corresponds more closely to the way how planning problems are solved by means of search-based techniques. As a consequence, methods and results from classical planning are more easily exploitable, such as techniques for partial order, regression planning, least commitment, as well as control strategies to guide the search. As a matter of fact the original motivation of this work was to give a "logical rewriting" to assertions in the DDL.1 specification language [4][1] and the model search method roughly corresponds to the planning algorithm given for DDL.1.

The paper is organised as follows. Next section shortly describes the core of our approach, that is, viewing plan generation as model building and using tableaux for linear temporal logic for this purpose. Section 3 introduces the syntax and semantics of the temporal logic TEMPOREP and Section 4 presents the prefixed tableau calculus for it, that enjoys a form of soundness and completeness w.r.t. finite model construction. Section 5 concludes the work.

2 Plan generation as model search via tableaux

Model construction in temporal logic is the main concern of *executable temporal logics*. These are based on an imperative reading of logic, where executing a temporal specifica-

[1] As a first step, the quantitative aspects of that work have been abstracted away.

tion means constructing a model for it [6]. The executable temporal language METATEM [1] 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. In a planning setting, this reading of formulae and their execution techniques correspond to a form of total-order, forward-chaining planning [2].

We introduce a temporal language based on discrete, linear time, whose modal part is a sub-language of propositional METATEM, and, similarly to [1], formulate an execution mechanism based upon model construction techniques related to analytic tableaux. We show that regression planning can be accomplished in what can be called a "declarative future implies imperative past" model. In other terms, a goal, which is a formula of the form $\Diamond A$ is at first executed by the creation of a time point t (greater or equal to the starting time point) where A is true. Then, the presently considered time becomes that time point t and the interval in between the starting point and t is built up by regression, applying axioms of the form "in order to have A true, it must be the case that previously B has been true". In this sense we speak of "declarative future implies imperative past": once t is considered the present time, the history before it is built up accordingly.[2]

Model building can be accomplished by means of a tableau-like construction: starting from the set of formulae describing the initial state, the problem constraints and a suitable description of the goal, model building amounts to the search of a complete, though finite open branch in a tableau. A tableau is a tree, whose nodes are labelled by formulae, that is built by a stepwise construction, expanding formulae in its nodes as described by a set of rules. Intuitively, the construction of a tableau amounts to the systematic search for a model of the initial formula or set of formulae. When tableau systems are used for automatic deduction purposes, proving the validity of A is reduced to proving that $\neg A$ is unsatisfiable. And this happens to be the case just when the systematic search for models of A fails. Conversely, when all the formulae in a tableau branch have been expanded (in that case the branch is called *complete*), if the branch is not contradictory, then it actually represents a partial model of the initial set of formulae. Note that tableau methods have proven to be efficient reasoning methods for practical purposes when proofs are guided by control knowledge that help keep the search space to a manageable size.

The tableau method proposed in this work is based on the use of prefixes that, intuitively, label each formula as being true in a particular state, in the style of [7, 8]. The calculus allows for partial ordering of time points, by means of a set of *temporal constraints* that is built up step by step as the construction of a tableau branch proceeds. A tableau branch is considered "complete" only when the time constraints establish a total ordering of time points. Hence, this reflects a form of plan-space search planning.

It is well known that the intrinsic complexity of the decision problem for propositional linear temporal logic is high [11]. However, most planning domains can be described by use of a syntactically restricted class of temporal formulae. Hence, although the issue deserves further investigation, it seems reasonable to devise specific search strategies, able to deal with plan generation, whose complexity falls below the general complexity of linear temporal logic.

[2] Obviously, reversing the point of view, i.e. taking the goal as a starting point and the initial situation as a time point in the future, would also allow a form of regression planning in METATEM.

3 The temporal logic TEMPOREP

The temporal language of TEMPOREP ("TEMporal logic for Partial Order, REgression Planning") takes into account the fact that, in the description of most planning domains, variables that may change values across time are to be considered. Actions can themselves be considered as relevant examples of such *state variables*. However, we assume that the set of values a single state variable is allowed to assume is finite. Besides assertions about the values of state variables, ordinary relational assertions are allowed. However, the set of term tuples that can be arguments of each relation symbol is finite too. Therefore, the language is a kind of sorted first-order logic over finite domains. Alternatively, it can be seen as a propositional language (it has in fact the computational properties of propositional temporal logic), where the use of variables and quantifiers are abbreviations used for notational convenience, allowing also a more efficient use of tableau methods (by exploiting unification) and reflecting similarities with the least commitment approach.

A language $\mathcal{L}$ of the temporal logic TEMPOREP is built over a denumerable set X of ordinary variables, a finite set of state variables $SV = \{sv_1, sv_2, \ldots, sv_k\}$, a finite set F of constants and function symbols with associated arity, a finite set R of n-ary relation symbols, the equality predicate $=$, the propositional operators $\neg, \wedge, \vee$ and the modal operators: $\Box$ (always in the future), $\Diamond$ (sometime in the future), $\bigcirc$ (weak next), $\overleftarrow{\bigcirc}$ (weak last), $\odot$ (strong next) and $\overleftarrow{\odot}$ (strong last). We choose to take all the modal operators as primitive and negation over non-atomic formulae as a defined symbol, in order to preserve compactness in the formulation of the tableau rules, without hindering the reader not accustomed with Fitting's notation [7]. As a further advantage, we need not use the notion of positive and negative occurrences of a sub-formula.

Every state variable, function and relation symbol has associated a *domain*, i.e. a *finite* set of ground terms: for $sv \in SV$, $f \in F$, $p \in R$, the definition of the language $\mathcal{L}$ specifies: $domain(sv) = \{M_1, ..., M_n\}$, $domain(f) = \{\overline{M_1}, ..., \overline{M_n}\}$, $domain(p) = \{\overline{M_1}, ..., \overline{M_n}\}$, where each M_j is a ground term and $\overline{M_i}$ a tuple of ground terms. The state variables in TEMPOREP are treated as *non-rigid designators*, i.e. their interpretation may vary from state to state. The interpretation of constants and function symbols is rigid. *Ground rigid terms* are built as usual out of constants and function symbols in F, but limited by the respective domains:

- if $f \in F$ is an n-ary function symbol and $\langle M_1, ..., M_n \rangle \in domain(f)$, then $f(M_1, ..., M_n)$ is a ground rigid term;

Therefore, the set of ground rigid terms is finite. It will be denoted by $T_{\mathcal{L}}$. Ground rigid terms denote values that may be assumed by the state variables. The domain associated to a state variable specifies which terms are allowed to be values for it.

Ground literals are defined as follows.

- If $sv \in SV$ and M is a ground rigid term in the domain of SV, then $sv = M$ and $\neg(sv = M)$ (abbreviated by $sv \neq M$) are literals;
- if $p \in R$ is an n-ary relation symbol and $\langle M_1, ..., M_n \rangle \in domain(p)$, then $p(M_1, ..., M_n)$ and $\neg p(M_1, ..., M_n)$ are literals.

Modal formulae are built out of literals by use of $\wedge$, $\vee$, and the modal operators. They are defined inductively, by means of the usual clauses for propositional operators and:

- If A is a formula, then $\Diamond A$, $\bigcirc A$, $\overleftarrow{\bigcirc} A$, $\odot A$ and $\overleftarrow{\odot} A$ are formulae.
- If A is a formula and A' results from A by replacing some occurrences of ground terms (possibly none) with ordinary variables in X, then $\Box A'$ is a formula.

Implication $\rightarrow$ and negation over non-atomic formulae are defined as usual. In particular: $\neg\Box A =_{Def} \Diamond\neg A$, $\neg\Diamond A =_{Def} \Box\neg A$, $\neg\bigcirc A =_{Def} \odot\neg A$, $\neg\odot A =_{Def} \bigcirc\neg A$, $\neg\overleftarrow{\bigcirc} A =_{Def} \overleftarrow{\odot}\neg A$, $\neg\overleftarrow{\odot} A =_{Def} \overleftarrow{\bigcirc}\neg A$.

Free ordinary variables are intended as universally quantified. However, since the domain associated to every state variable and relation symbol is finite, a universal statement (i.e. a formula with free variables) can be considered as the abbreviation for the conjunction of all its finite instances. Note that ordinary variables from X are allowed to occur only inside statements of "necessary force". This restriction is not essential in a logical view, but it can be exploited in the definition of strategies for model building reflecting ordinary planning practice. Although also the use of "existential variables" could improve the search procedure, we do not consider this feature here.

A *temporal structure* T is a finite sequence of states $\langle t_0, ..., t_k\rangle$, in which all the states are distinct. Since a temporal structure is isomorphic to an initial segment of the natural sequence $0, 1, ...$, the first state in T (t_0) will be sometimes denoted by 0, and, if t_i, t_j are consecutive states in T, then we abuse notation and write $t_j = t_i + 1$, $t_i = t_j - 1$. The elements of T are called *states*. Interpretations are taken to be Herbrand models: their domain is the set of ground rigid terms of the language and the interpretation of ground rigid terms is the identity on $T_{\mathcal{L}}$. If $\mathcal{L}$ is a TEMPOREP-language, an $\mathcal{L}$-interpretation $\mathcal{M}$ is a triple $\langle T, s, \sigma\rangle$, where:

- T is a temporal structure;
- s a function $s : T \rightarrow (SV \rightarrow T_{\mathcal{L}})$. I.e., for any $t \in T$, $s(t)$ is a function assigning a ground rigid term to each state variable sv_i;
- σ is a function on states, providing an interpretation to the relation symbols in R for any state in T. I.e. if $t \in T$ and $p \in R$ is an n-ary relation symbol, then $\sigma(t)(p) \subseteq T^n_{\mathcal{L}}$.

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

1. $\mathcal{M}_t \models sv = M$ iff $s(t)(sv) = M$.
2. $\mathcal{M}_t \models p(M_1, ..., M_n)$ iff $\langle M_1, ..., M_n\rangle \in \sigma(t)(p)$.
3. $\mathcal{M}_t \models \neg L$ iff $\mathcal{M}_t \not\models L$, if L is a (positive) literal.
4. $\mathcal{M}_t \models A \wedge B$ iff $\mathcal{M}_t \models A$ and $\mathcal{M}_t \models B$.
5. $\mathcal{M}_t \models A \vee B$ iff either $\mathcal{M}_t \models A$ or $\mathcal{M}_t \models B$.
6. $\mathcal{M}_t \models \Box A$ iff for all $t' \geq t$ and for every ground instance A' of A, $\mathcal{M}_{t'} \models A'$.
7. $\mathcal{M}_t \models \Diamond A$ iff there exists $t' \in T$ such that $t' \geq t$ and $\mathcal{M}_{t'} \models A$.
8. $\mathcal{M}_t \models \bigcirc A$ iff either t is the last state in T or $\mathcal{M}_{t+1} \models A$.

9. $\mathcal{M}_t \models \overleftarrow{\bigcirc} A$ iff either t is the initial state in T ($t = 0$) or $\mathcal{M}_{t-1} \models A$.

10. $\mathcal{M}_t \models \odot A$ iff $t + 1 \in T$ and $\mathcal{M}_{t+1} \models A$.

11. $\mathcal{M}_t \models \overleftarrow{\odot} A$ iff $t \neq 0$ and $\mathcal{M}_{t-1} \models A$.

A formula A is true in $\mathcal{M}$ ($\mathcal{M} \models A$) iff $\mathcal{M}_0 \models A$, i.e. A is true in the first state 0. If $\mathcal{M} \models A$ then $\mathcal{M}$ is a model of A. Truth of sets of formulae is defined as usual.

Planning domains that can be specified by use of a set of state variables, their associated values and a set of constraints of the form *MEETS*, *MET-BY* and *DURING* (like in [4]) can be easily described in *TEMPOREP*. A suitable choice of the language allows the explicit treatment of actions, by reification. These can in fact be considered as the values of a particular state variable, that we call *action*. The value of *action* in a state t represents the action determining the transition to the next state. The inclusion of relation symbols in the language allows one to treat atomic statements that are different from state variable value assertions, and therefore to speak of action preconditions and effects.

As a short example, let us assume that our goal is being drunk and yet have wine at our disposal. The actions we can perform are buying wine and drink. The planning domain can be described by use of a language consisting of the single state variable *action*, having values in $\{drink, buy\_wine\}$. The propositional letters (0-ary relation symbols) *drunk* and *wine* represent, respectively, being drunk and possessing wine. The TEMPOREP theory describing this domain contains the following formulae:

$$
\begin{array}{ll}
(c1) & \neg drunk \wedge \neg wine \\
(c2) & \Diamond(drunk \wedge wine) \\
(c3) & \Box(drunk \rightarrow \overleftarrow{\bigcirc}(action = drink \vee drunk)) \\
(c4) & \Box(wine \rightarrow \overleftarrow{\bigcirc}(action = buy\_wine \vee (wine \wedge action \neq drink))) \\
(c5) & \Box(action = drink \rightarrow wine)
\end{array}
$$

$c1$ and $c2$ are the description of the initial state and goal, respectively. $c5$ expresses a precondition requirement for the action *drink*, while $c3$ and $c4$ play the role of "action description axioms". Remarkably, in the setting of goal-driven model construction for temporal formulae, only one half of the usual action description axioms is required: in deductive planning a double implication in $c3$, $c4$ is needed. The intuitive reason why we can dispense with the other half of the double implication is strictly related to the difference between deduction and model building by "regression": when we try to build a model for A, we are free to assume anything we need in order to make A true. Therefore the domain description axioms only have to take care that no *ad hoc*, unjustified assumption is made. Rather than *frame axioms*, we need statements ensuring that, if the truth value of A changes across time, then one of the actions having effect on A has occurred exactly when A changed its truth value.

4 A calculus for finite model construction

Finding a plan leading from the actual state to a goal state amounts to the construction of a temporal model $\mathcal{M}$ satisfying the specification of the planning domain.

In this section we present a tableau system for TEMPOREP and state some of its properties. We use prefixed unsigned tableaux with temporal constraints, in the style of [8]. Prefixes are symbols intending to denote the states of a temporal sequence. A node in a tableau is labelled by a prefix and a formula, $t : A$, and its intuitive meaning is that A is true in state t. Every branch $\mathcal{B}$ in a tableau is associated with a set of *temporal constraints* $\delta(\mathcal{B})$, i.e. expressions of the form $t_1 = t_2$, $t_1 < t_2$, $t_1 \leq t_2$, or $t_1 = succ(t_2)$, where t_1, t_2 are prefixes occurring in the branch. The role of $\delta(\mathcal{B})$ is to describe the partial order on the set of time points (prefixes) occurring in the branch. When adding a node to the end of a branch, $\delta(\mathcal{B})$ is either added new constraints or left unchanged.

In order to solve a planning problem, an initial tableau is built containing a single branch with the formulae describing the domain, the initial state and the goals, all with the same prefix, called the *initial prefix*. The set of temporal constraints is initially empty. The tableau is expanded by choosing a node and adding its *expansions* to the end of each branch passing through that node, such as defined by means of a set of *expansion rules*. In the formulation of the rules, the expansions of a node (that appears above the derivation sign) are one or more assertions of the form $t : A \;||\; \Delta$ (below the derivation sign). When a node is expanded by application of one of such rules, every branch passing through the node is expanded by addition of new nodes and by updating the associated set of temporal constraints. When the rule has a single expansion $t : A \;||\; \Delta$, then $t : A$ is added at the end of the branch and the set of temporal constraints associated with the branch is added the temporal constraints in Δ. When there are two expansions, written one below the other, the branch is added two nodes, one below the other. When the expansions $t_i : A_i \;||\; \Delta_i$ are written one beside the other (in *branching rules*), the application of the rule generates a fork in the branch, each fork having one of the $t_i : A_i$ at its end and the set of temporal constraints associated to the corresponding branch is added Δ_i. For convenience, we omit to write the set Δ in the rules when it is the empty set.

We assume here that a correct and complete (w.r.t. the standard interpretation over the natural numbers), as well as decidable, deduction system for temporal constraints is defined, and write $\Delta \vdash \tau$, where τ is a temporal constraint, to mean that τ is derivable from the set of constraints Δ.

Definition 1 *If $\mathcal{B}$ is a branch in a tableau, then $\delta(\mathcal{B})$ is* inconsistent *if $\delta(\mathcal{B}) \vdash t < t$ for some prefix t occurring in $\mathcal{B}$, and it is* total *whenever it induces a total order, i.e. for every two prefixes t_1, t_2 occurring in $\mathcal{B}$, either $\delta(\mathcal{B}) \vdash t_1 = t_2$, or $\delta(\mathcal{B}) \vdash t_1 < t_2$, or $\delta(\mathcal{B}) \vdash t_2 < t_1$.*

The tableau rules include the classical rules, where $\delta(\mathcal{B})$ is not modified:

$$(\alpha)\quad \frac{t : A \wedge B}{\begin{array}{l} t : A \\ t : B \end{array}} \qquad (\beta)\quad \frac{t : A \vee B}{t : A \qquad\qquad t : B}$$

The modal rules are the following:

$$(\nu)\quad \frac{t : \Box A}{t' : A'} \qquad (\pi)\quad \frac{t : \Diamond A}{t'' : A \;||\; \{t \leq t''\}}$$

where A' is a ground instance of A (ν-rule) and t'' is a new prefix (π-rule). The ν-rule additionally requires that $\delta(\mathcal{B}) \vdash t \leq t'$.

$$(wnext)\quad \frac{t : \bigcirc A}{t' : A \;\;||\;\; \{t' = succ(t)\}} \qquad (wlast)\quad \frac{t : \overleftarrow{\bigcirc} A}{t' : A \;\;||\;\; \{t = succ(t')\}}$$

$$(next)\quad \frac{t : \odot A}{t' : A \;\;||\;\; \{t' = succ(t)\}} \qquad (last)\quad \frac{t : \overleftarrow{\odot} A}{t' : A \;\;||\;\; \{t = succ(t')\}}$$

The *wnext* rule requires that for some t'', $\delta(\mathcal{B}) \vdash t < t''$ and *wlast* that for some t'', $\delta(\mathcal{B}) \vdash t'' < t$. In all these four rules, t' is a new prefix.

This concludes the set of *logical rules*. In order to keep the formulation of the rules as compact as possible, we do not mention here the use of unification, that can replace the implicit instantiation in the ν-rule in the obvious way.

The last rule manipulates temporal constraints. It is not applied to any particular node in the branch, rather it updates the set of temporal constraints associated to the branch. If $\mathcal{B}$ is the branch, its application generates three branches from $\mathcal{B}$:

$$(sync)\quad \frac{\mathcal{B}}{||\;\; \{t = t'\} \qquad ||\;\; \{t < t'\} \qquad ||\;\; \{t' < t\}}$$

Each of them has no new node, but the associated set of temporal constraints is added, respectively, $t = t'$, $t < t'$, $t' < t$, where t and t' are prefixes occurring in the branch. The rule is used in order to build a total order on prefixes. In particular (leftmost branch), it allows one to identify states, thus performing a form of synchronisation.

In the following definition, we informally write $t : A \in \mathcal{B}$ to mean that $t : A$ occurs in $\mathcal{B}$, and $[t] : A \in \mathcal{B}$ iff $t' : A \in \mathcal{B}$ for some t' such that $\delta(\mathcal{B}) \vdash t = t'$.

Definition 2 *Let $\mathcal{B}$ be a tableau branch.*

1. *A prefix t is* completely expanded in $\mathcal{B}$ *iff the following conditions hold:*
   - *for every $t : \Box A \in \mathcal{B}$ and for every t' such that $\delta(\mathcal{B}) \vdash t \leq t'$ and for every ground instance A' of A, $[t'] : A' \in \mathcal{B}$;*
   - *if $t : A \wedge B \in \mathcal{B}$ then $[t] : A \in \mathcal{B}$ and $[t] : B \in \mathcal{B}$;*
   - *if $t : A \vee B \in \mathcal{B}$ then either $[t] : A \in \mathcal{B}$ or $[t] : B \in \mathcal{B}$;*
   - *if $t : \Diamond A \in \mathcal{B}$, then for some t' such that $\Delta \vdash t \leq t'$, $t' : A \in \mathcal{B}$;*
   - *if $t : \bigcirc A \in \mathcal{B}$ and, for some t', $\Delta \vdash t < t'$, then there exists t'' such that $t'' : A \in \mathcal{B}$ and $\Delta \vdash t'' = succ(t)$;*
   - *if $t : \overleftarrow{\bigcirc} A \in \mathcal{B}$ and t is not the initial prefix, then there exists t' such that $t' : A \in \mathcal{B}$ and $\Delta \vdash t = succ(t')$;*
   - *if $t : \odot A \in \mathcal{B}$, then there exists t' such that $t' : A \in \mathcal{B}$ and $\Delta \vdash t' = succ(t)$;*
   - *if $t : \overleftarrow{\odot} A \in \mathcal{B}$, then there exists t' such that $t' : A \in \mathcal{B}$ and $\Delta \vdash t = succ(t')$.*
2. *A branch $\mathcal{B}$ is* closed *iff one of the following conditions hold:*
   (a) *the set $\delta(\mathcal{B})$ is inconsistent;*
   (b) *$\mathcal{B}$ contains two nodes of the form $t : A$ and $t' : \neg A$ where $\delta(\mathcal{B}) \vdash t = t'$;*

(c) $\mathcal{B}$ *contains two nodes of the form* $t : sv = M$ *and* $t' : sv = N$, *where* $\delta(\mathcal{B}) \vdash t = t'$ *and* M *and* N *are different ground terms;*

(d) *if* t_0 *is the prefix used to initialize the tableau, there is a node* $t' : A$ *in* $\mathcal{B}$ *with* $\delta(\mathcal{B}) \vdash t' < t$.

$\mathcal{B}$ *is open if it is not closed.*

3. $\mathcal{B}$ *is* complete *iff every prefix occurring in* $\mathcal{B}$ *is completely expanded and* $\delta(\mathcal{B})$ *is total.* $\mathcal{B}$ *is a* model branch *iff it is open and complete.*

Intuitively, a model branch should describe a partial model of the initial set of formulae. For example, a prototype implementation of the tableau system for TEMPOREP, run on the planning problem of our previous example, generated a branch, representing a solution, with four equivalence classes of prefixes, $[t_1], .., [t_4]$, where $t_i = succ(t_{i-1})$ (for $i = 2, .., 4$), and the set of literals associated to each of such classes is:

$$[t_1] : action = buy\_wine, \neg drunk, \neg wine, action \neq drink$$
$$[t_2] : wine, action = drink, \neg drunk$$
$$[t_3] : drunk, action = buy\_wine, \neg wine, action \neq drink$$
$$[t_4] : drunk, wine, action \neq drink$$

The tableau calculus for TEMPOREP is complete w.r.t. model building and a form of soundness holds. Remark that, in the context of model construction, the roles of soundness and completeness are reversed, i.e. what should amount to a completeness result for deduction purposes is what we need in order to establish the soundness of the calculus for model building purposes, and *vice versa*.

Under a suitable definition of satisfiability of tableau branches in a given temporal interpretation (details can be found in [3]), it can in fact be shown that, if S is a set of formulae that has a model $\mathcal{M}$, then any tableau for S has a branch $\mathcal{B}$ that is satisfiable in $\mathcal{M}$. Moreover, there exists a tableau for S that has a finite model branch that is satisfiable in $\mathcal{M}$ (completeness w.r.t. model construction).

The calculus is not complete for deduction purposes, unless either an upper bound to the length of branches is set or a suitable loop detection mechanism is included. In fact, there are unsatisfiable sets of formulae that have no closed tableau. This is due to the fact that infinite open branches may have no model. Consider for example the set $\{q, \Box\overleftarrow{\bigcirc}\neg q, \Diamond\neg q\}$, that is unsatisfiable in a discrete temporal frame. Any tableau for it consists of a single infinite branch, that has no discrete model (in the context of model building, this turns into an unsoundness result: there may be infinite model branches that have no model). For this reason, tableau procedures for discrete, linear temporal logics are usually equipped with some sort of loop detection mechanism (see, for example, [1, 13]). In addition, the completeness issue is particularly involved in the presence of temporal constraints, as pointed out in [8].

However, in the planning perspective, we may take advantage from the fact that the interest is only in *finite* models, so that, in practice, there is no need to distinguish between infinite unsatisfiable branches and infinite satisfiable ones. For this reasons, the following weak form of soundness is of interest: if S is a set of TEMPOREP formulae, $\mathcal{T}$ a tableau for S and $\mathcal{B}$ a finite model branch in $\mathcal{T}$, then there exists a (finite) temporal interpretation $\mathcal{M}$ such that $\mathcal{B}$ is satisfiable in $\mathcal{M}$. Therefore, if we agree that a solution to a planning

problem $\mathcal{P}$ specified by the set of formulae $S_{\mathcal{P}}$ is any finite model of $S_{\mathcal{P}}$, then $\mathcal{P}$ has a solution $\mathcal{M}$ if and only if there exists a tableau for $S_{\mathcal{P}}$ with a finite model branch satisfiable in $\mathcal{M}$.

A decision procedure can be obtained by exploiting the fact that if S has a finite model, then it has a model $\mathcal{M}$ with no more than $2^{|S|}$ states (similarly to Lemma 4.5 in [11]) and the number of prefixes that have to be expanded in a tableau branch satisfiable in $\mathcal{M}$ can be bounded in function of the number of states in $\mathcal{M}$. In [3], a provably correct and complete decision procedure for finite model construction is described, based on the construction of a single tableau, suitably constrained so that it is bound to terminate and return a finite model of the initial set of formulae, if there exists one.

Furthermore, we remark that, in practical planning problems, we look not only for a "finite" solution, but for a plan that can be carried out in a reasonable amount of time, that is usually far from reaching the above exponential upper bound. This means that setting a more effective bound on the maximal depth a tableau branch is allowed to reach is a fair and admissible solution for practical purposes. The above mentioned decision procedure for finite model construction can take a maximal size k for the model to be built as additional input, and returns a model with no more than k states (if there exists one).

5 Conclusions

This work is based on the view of plan generation as the search for finite models of plan specification logical formulae. A modal temporal language is introduced for the representation of planning domains, based on a discrete, linear model of time. A calculus for solving planning problems (i.e. building finite models of temporal formulae) is defined, based on prefixed analytic tableaux, and reflecting planning methodologies such as regression, partial order, least commitment. Grounded on the calculus, a provably sound and complete decision procedure for finite model construction can be defined, that is parametric w.r.t. the maximal time length a plan is allowed to reach. The described approach takes advantage of significant results from classical planning, while furnishing a tool for the description and solution of planning problems that relies on a stable formal framework.

The language has a rich expressive power, allowing the specification of very complex planning tasks, including the behaviour of physical domains in situations similar to the ones addressed with the DDL.1 description language. The focus of our current research efforts is the problem of inference control. A related issue is the study of a general form for plan specification formulae, that, still allowing to model a large class of problems, would reduce the complexity of the associated search procedure and facilitate the possibility of exploiting classical techniques for search control.

Directions for future investigation concern a further enrichment of the expressive power of the language (for example, the use of the *Since* operator would highly increase the degree of freedom in building partial plans; moreover, time points can be stretched into intervals with an associated duration, thereby taking into account the quantitative aspect expressible in some planning languages), and the creation of user supporting tools for domain definition with TEMPOREP.

References

[1] 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.

[2] 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.

[3] S. Cerrito and M. Cialdea Mayer. A prefixed tableau calculus for plan generation in linear temporal logic. Technical Report RT-DIA-24-1997, Dipartimento di Informatica e Automazione, Università di Roma Tre, 1997.

[4] 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.

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

[6] 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.

[7] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel Publishing Company, 1983.

[8] 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.

[9] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 359–363. John Wiley & Sons, Ltd., 1992.

[10] 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.

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

[12] B. Stephan and S. Biundo. Deduction-based refinement planning. In *AIPS-96*, pages 213–220, 1996.

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