First Order Linear Temporal Logic over Finite Time Structures

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

[1] Université de Paris-Sud, L.R.I., e-mail: {serena, praud}@lri.fr
[2] Università di Roma Tre, e-mail: cialdea@dia.uniroma3.it

Abstract. In this work, the notion of provability for first order linear temporal logic over *finite* time structures, $FO\text{-}LTL^{fin}$, is studied. We show that the validity problem for such a logic is not recursively enumerable, hence $FO\text{-}LTL^{fin}$ is not recursively axiomatizable.
This negative result however does not hold in the case of *bounded* validity, that is truth in all temporal models where the object domain is possibly infinite, but the underlying sequence of time points does not exceed a given size. A formula is defined to be k-valid if it is true in all temporal models whose underlying time frame is not greater than k, where k is any fixed positive integer. In this work a tableau calculus is defined, that is sound and complete with respect to k-validity, when given as input the initial formula and the bound k on the size of the temporal models. The main feature of the system, extending the propositional calculus defined in [7], is that of explicitly denoting time points and having tableau nodes labelled by either expressions intuitively stating that a formula holds in a given temporal interval, or "temporal constraints", i.e. linear inequalities on time points. Branch closure is reduced to unsatisfiability over the integers of the set of temporal constraints in the branch.

1 Introduction

The model of time underlying Linear Temporal Logic (LTL) is a discrete, linear sequence of states, usually taken to be bounded in the past and infinite in the future. In other words, the set of time points is isomorphic to $\mathbb{N}$. Different sets of temporal operators may be considered: mainly, future time operators ($\Box$: always, $\Diamond$: eventually, $\bigcirc$: next, $\mathcal{U}$: until), possibly restricting to the fragment with $\Box$ and $\Diamond$ only, or both past and future time ones ("full" LTL). In the propositional case, several sound, complete and terminating proof-systems for LTL have been provided; see for instance [19, 18, 4, 2]. As far as first order LTL, equipped with the infinite semantics, is concerned, it has been proved that no complete recursive axiomatization can exist (see [14, 1, 16]). This explains why relatively little work has been done on the proof theory of first order LTL. An exception is [2], where a first order non-clausal resolution system is defined, which is complete w.r.t. the class of the so called "formulae provable with a clock".

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

In this work, we consider first order LTL with a *finite* semantics, FO-LTL^{fin}, where time is assumed to be finite both in the past and in the future, while the object domain can be infinite. Its interest is due to the fact that there are problems in Computer Science and Artificial Intelligence where only a finite fragment of the time sequence is of interest. For instance, in the case of a database evolving through time by means of updates, queries involving several states of the database, as well as dynamic integrity constraints, may be expressed by first order temporal formulae and the database history, from the initial state up to the current state, can be modeled by a finite time temporal interpretation [11, 3, 6]. Similarly, the specification of a planning problem may be expressed by means of a set of temporal logic formulae; since a plan is a finite sequence of actions, leading from the initial situation to the desired goal, it can be modeled by a finite time temporal interpretation [7, 8].

In [7] the propositional version of LTL over finite time frames is studied and a labelled tableau calculus is defined and proved sound, complete and terminating. Obviously, however, propositional logic is expressively too poor for the above mentioned applications. In the case of FO-LTL^{fin}, decidability is clearly lost, but what about semi-decidability? In both [1] and [16], the kernel of the proof techniques used to get the intrinsic incompleteness result for first order LTL with the infinite semantics is the possibility of encoding arithmetic into such a logic. It is apparent that such proofs cannot be adapted to the case of FO-LTL^{fin}, where a time frame is isomorphic to a *finite* initial segment of the natural numbers. In this work, however, we show that the validity problem for FO-LTL^{fin} is still not recursively enumerable, by use of a different, and quite natural, technique. As a consequence, no effective proof system for FO-LTL^{fin} may exist.

In the case of FO-LTL^{fin}, a weaker notion than validity can be considered. If k is any given positive integer, the notion of k-validity of a formula is defined as follows: F is k-valid iff it is true in any finite interpretation whose underlying time frame is not greater than k. Such a notion of *bounded validity* is still a useful one in several applications, for instance, in databases, where the question of the preservation of an integrity constraint after a given number k of updates is important [5], and in planning, where the searched plan can be reasonably required not to exceed a given length.

In this work we propose a labelled tableau calculus which is sound and complete w.r.t. the notion of bounded FO-LTL^{fin}-validity. The main difference of our approach, w.r.t. to "traditional" tableaux for LTL, is the use of *labels* on formulae, making it possible to embed semantical information in the calculus itself, namely the fact that a temporal frame is isomorphic to $\mathbb{N}$. Tableaux rules introduce linear constraints on world variables, so that specific algorithms for solving integer constraints may be used in order to check satisfiability. The idea of using linear constraints over integers in proof systems for propositional linear temporal logic (with the infinite semantics) first appeared in [13] and [18, 17], and it has been revisited in the calculus proposed in [7, 8]. In the present work, we extend the calculus of [7] to the first order case, obtaining a system that is sound and complete w.r.t. bounded validity (albeit sound, modulo a minor modification, but obviously incomplete, w.r.t. validity in finite time frames *tout court*).

2 Syntax and Semantics of $FO\text{-}LTL^{fin}$

The language of linear temporal logic we consider includes the classical operators $\neg, \wedge, \vee$, $\bot$ (always false), $\top$ (always true), $\forall$ and $\exists$, the unary modal operators $\Box$ (always in the future), and $\overleftarrow{\Box}$ (always in the past), and the binary ones $\mathcal{U}$ (until) and $\mathcal{S}$ (since). The alphabet contains a set P of predicate symbols and a set F of function symbols; all the function symbols in F are *rigid*.[1] When the distinguished equality predicate $=$ is considered, it is also taken to be rigid, in the sense that its interpretation (identity) does not change over time.

The semantics of the language with predicate symbols in P and function symbols in F is defined as follows. A *temporal frame* T is a finite initial segment of the natural numbers, $\langle 0, \cdots, k\rangle$; its elements are called *time points*. A temporal interpretation $\mathcal{M}$ is a quadruple $\langle T, D, \delta_P, \delta_F\rangle$ where T is a temporal frame, D is a non-empty set (the object domain, possibly infinite), δ_P is a mapping from $P \times \{0, ..., k\}$ to the set of relations on D such that, for any n-ary predicate symbol p with $n > 0$ and any $i \in T$, $\delta_P(p, i)$ is a subset of D^n, and δ_F is a mapping from F to the set of (total) functions on D such that, for any n-ary function symbol f with $n \geq 0$, $\delta_F(f)$ is a function from D^n to D.

Let σ be a variable assignment, i.e. a function from the set of variables Var to the domain D. Given any term t, its value $[t]_{\mathcal{M},\sigma}$ w.r.t. the interpretation $\mathcal{M}$ and the variable assignment σ, is defined like in the classical case. Note that, since the language is assumed to have only rigid functional symbols, the interpretation of a term is the same at any time point.

If A is a formula, the satisfiability relation $\mathcal{M}_{\sigma,i} \models A$ (to be read: A is satisfied by $\mathcal{M}$ w.r.t. the variable assignment σ at the time point i) extends the classical definition – with base cases $\mathcal{M}_{\sigma,i} \models p(t_1, ..., t_n)$ iff $\langle [t_1]_{\mathcal{M},\sigma}, ..., [t_n]_{\mathcal{M},\sigma}\rangle \in \delta_P(p, i)$, and a similar one treating equality – as follows:

1. $\mathcal{M}_{\sigma,i} \models \Box A$ iff for all j such that $i < j$, $\mathcal{M}_{\sigma,j} \models A$.
2. $\mathcal{M}_{\sigma,i} \models A\mathcal{U}B$ iff $\exists j \in T$ such that $i < j$ and $\mathcal{M}_{\sigma,j} \models B$, and for any k with $i < k < j$ $\mathcal{M}_{\sigma,k} \models A$.
3. $\mathcal{M}_{\sigma,i} \models \overleftarrow{\Box} A$ iff for all $j < i$, $\mathcal{M}_{\sigma,j} \models A$.
4. $\mathcal{M}_{\sigma,i} \models A\mathcal{S}B$ iff $\exists j \in T$ such that $j < i$ and $\mathcal{M}_{\sigma,j} \models B$ and for any k with $j < k < i$ $\mathcal{M}_{\sigma,k} \models A$.

Note that, due to the "strict" interpretation of the modal operators (excluding the present time point in both cases of future and past time operators), the weak and strong "Next" and "Last" operators are definable. The $\Diamond$ (eventually) and $\overleftarrow{\Diamond}$ (sometimes in the past) operators are also definable in terms of,

[1] Note that non-rigid function symbols may always be simulated via predicates. Thus, considering only rigid function symbols does not affect the expressive power of the considered logic.

respectively, $\mathcal{U}$ and $\mathcal{S}$, as well as all the other standard temporal operators. The operators taken as primitive are all necessary in order to have a negation normal form property, that is exploited in Section 4.

When A is closed, we omit σ and we just write $\mathcal{M}_j \models A$ to mean that A is satisfied by the structure $\mathcal{M}$ at the time point j. Truth of a formula A is satisfiability of its universal closure in the initial state: a formula A is true in $\mathcal{M}$ (and $\mathcal{M}$ is a model of A) iff $\mathcal{M}_0 \models \forall x_1, ..., \forall x_n A$, where $x_1, \cdots, x_n$ are all the free variables in A. Truth of sets of formulae is defined as usual. Satisfiability of a closed formula is truth in at least one model, validity is truth in all models.[2] If k is a positive integer and A a closed formula, A is said to be k-satisfiable if it is true in some model whose temporal frame $\langle 0, ..., m\rangle$ is such that $m \leq k$. A is k-valid iff it is true in any model whose temporal frame $\langle 0, ..., m\rangle$ is such that $m \leq k$.

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.

3 $FO\text{-}LTL^{fin}$ is not Semi-decidable

In [1] it is proved that a complete recursive axiomatization for first order LTL, equipped with the infinite semantics, cannot exist, by exploiting the power of the future time temporal operators $\bar{\Box}$ (*non strict always*, whose semantics includes the present time point) and $\bigcirc$ (next). The proof makes use of the notion of Π^1_1 formulae, that is formulae having the form $\forall R_1 \ldots \forall R_k \forall F_1 \ldots \forall F_{k'} B$, where B is some classical first order formula and $0, s, \leq, +, \times, R_1, \ldots, R_k, F_1, \ldots, F_{k'}$ are all the predicate and function symbols in B. The complexity class Π^1_1 includes all problems no harder than the truth problem for Π^1_1 formulae. What is shown is that the LTL validity problem is Π^1_1-complete. The kernel of the proof consists in exhibiting a recursive embedding E of Π^1_1 formulae in LTL formulae, such that, given any Π^1_1 formula F, F is true if and only if $E(F)$ is LTL valid. The definition of the mapping E uses the equality predicate $=$, one non-rigid constant a and only the (future) temporal operators $\bar{\Box}$, the corresponding "non strict eventually" operator and $\bigcirc$ (next) – of course, the use of a may be actually simulated by an appropriate use of predicate symbols. In [16] some interpretability results of classical arithmetical theories in temporal theories are proved and, as a corollary, two versions of the incompleteness result for first order LTL with the future time operators are obtained. Since, as it has been afterwards shown in [15], validity of first order LTL formulae containing $\bar{\Box}$ and $\bigcirc$ may be reduced to validity of formulae containing only $\bar{\Box}$ as a temporal

[2] A different definition of validity can also be found in the literature: A is valid iff for any temporal interpretation $\mathcal{M}$ and for any time point i, $\mathcal{M}_i \models A$. However, it is easy to see that the two notions of validity are interdefinable.

operator, it follows that validity for first order LTL with equality and $\bar{\Box}$ is not recursively enumerable.

In both [1] and [16], the kernel of the proof techniques used to get the incompleteness result is the possibility of encoding arithmetic into LTL. Thus, such proofs cannot be adapted to the case of $FO\text{-}LTL^{fin}$, where a time frame is isomorphic to a *finite* initial segment of the natural numbers.

However, it can be shown that the set of $FO\text{-}LTL^{fin}$ valid formulae is not recursively enumerable by reducing the complement of the halting problem for Turing Machines to the validity problem for $FO\text{-}LTL^{fin}$.

Theorem 1. *Given any Turing machine M and word w, a temporal formula $F_{M,w}$ can be constructed, such that $F_{M,w}$ is $FO\text{-}LTL^{fin}$-valid iff M never halts when given w as an input.*

Proof. The proof, whose details can be found in [9], bears some similarities with the proof of the fact that validity in finite domains of classical predicate calculus formulae is not recursively enumerable. However, in the case of classical logic, the set of finitely satisfiable formulae is trivially recursively enumerable, so that in order to conclude that "finite validity" is not so it suffices to show that finite satisfiability is not recursive, by exhibiting a formula, for each Turing Machine M and input word w, which is finitely satisfiable if and only if M halts on w. In the case of LTL^{fin}, the class of satisfiable formulae is not recursively enumerable either (since classical first order satisfiability can trivially be reduced to LTL^{fin} satisfiability), hence an explicit construction of a LTL^{fin} formula which is *valid* if and only if M does *not* halt on w must be accomplished.

The proof can be sketched as follows. Given a deterministic Turing machine :

$$M = \langle Q, \Sigma, \Delta, q_0, q_f \rangle$$

(where Q is the set of states, Σ the alphabet, Δ the transition table, q_0 the initial state and q_f the final state of the machine) and a word w in the alphabet of M, the language $\mathcal{L}_{M,w}$ of $F_{M,w}$ contains: a constant symbol 0 and a unary functional symbol s, the equality predicate and the binary relation symbol $<$, for each state $q_i \in Q$, the unary predicate symbol Q_i, and, for each symbol $s_i \in \Sigma$, the unary predicate symbol S_i.

We say that M halts before time k if for some time $n < k$, M is in state q_f at time n. We define a set of "intended interpretations" $\mathcal{M}^k$ of the language $\mathcal{L}_{M,w}$, for each $k \geq 0$ such that the machine M does not halt before time k, when given the word w as an input. The time structure in $\mathcal{M}^k$ is $\langle 0, ..., k \rangle$. The domain is $\mathbb{N}$. $\mathcal{M}^k$ assigns zero to 0 and the successor function to s. Moreover, if σ is a variable assignment and $t \in T$:

1. ${\mathcal{M}^k}_{\sigma,t} \models Q_i(x)$ iff at time t the machine is in state q_i, scanning square number $\sigma(x)$
2. ${\mathcal{M}^k}_{\sigma,t} \models S_i(x)$ iff at time t the symbol s_i is in square number $\sigma(x)$
3. ${\mathcal{M}^k}_{\sigma,t} \models x < y$ iff $\sigma(x)$ is less than $\sigma(y)$

Given a Turing machine M with input w, the formula $G_{M,w}$ is the conjunction of a set of formulae describing the behaviour of M (i.e. the initial state, the transition table of M, standard properties of s and $<$, the fact that the machine can be in at most one state at a time and that time ends when the final state is reached). The only temporal operators needed in the encoding are $\bar{\Box}$ (non strict always, including the present time point), defined as $\bar{\Box}F \equiv_{def} F \wedge \Box F$, and $\bigcirc$ (weak next), defined as $\bigcirc A \equiv_{def} \Box\bot \vee \bot \mathcal{U} A$.

The formula $\overline{H}_{M,w}$ is $\bar{\Box}\forall x \neg Q_f(x)$, intuitively stating, in the intended interpretations, that the machine never halts, and $F_{M,w}$ is $G_{M,w} \rightarrow \overline{H}_{M,w}$.

It can be proved that the formula $G_{M,w}$ logically implies a description D_t of time t, for each time point t such that the machine M has not halted before time t (i.e. for any $n = 0, ..., t-1$, M is not in the final state). Such formulae have the form $\bigcirc^{(t)}A$, where $\bigcirc^{(t)}$ is a t-length sequence of $\bigcirc$ operators and A describes the symbols on the tape and which the state of the machine and the scanned square are at time t.

The fact that if $F_{M,w}$ is valid then M never halts when given w as an input is straightforward. For the converse, let us assume that M never halts when given w as an input and however there exists a model $\mathcal{M}$ of $G_{M,w} \wedge \neg\overline{H}_{M,w}$. Let m be the size of its temporal frame. Since $\mathcal{M} \models \neg\bar{\Box}\forall x \neg Q_f(x)$, there is a time point $k \leq m$ such that $\mathcal{M}_k \models \exists x Q_f(x)$ (note that, if the usual encoding of Turing Machines in classical logic is considered, in the corresponding (classical) model, the time point k could be a non standard number). By using the fact that, if M never halts when given w as an input, $G_{M,w}$ logically implies some description D_k of state k, it can be proved that $G_{M,w}$ logically implies a closed formula of the form $\bigcirc^{(k)}Q_f(\overline{p})$, where $\overline{p}$ is some term of the form $s(s(...(0)...))$. Therefore $\mathcal{M}^k \models \bigcirc^{(k)}Q_f(\overline{p})$, where $\mathcal{M}^k$ is the intended interpretation of size k. It follows that at time k the machine M, when given w as an input, is in the final state q_f. I.e. M halts on w: contradiction. ∎

Such a result has an impact also on first order LTL with the *infinite* semantics. Consider the first order fragment LTL_{past}, containing only unary past time operators, and say that A is valid iff A holds at each time point of each temporal interpretation (in this case, the notion of validity defined in Section 2 is not interesting). A rather straightforward application of Theorem 1 enables us to prove the following result:

Theorem 2. *First order LTL_{past} is not recursively axiomatizable.*

4 First Order Bounded Tableaux

In this section, we propose a labelled tableau system to check k-satisfiability of $FO\text{-}LTL^{fin}$ formulae. It extends the system introduced in [7] for the propositional case, whose purpose is to verify whether a set of formulae in nnf is finitely satisfiable (the restriction to nnf formulae is introduced only to simplify the presentation of the rules). The main feature of the system, that is inspired by [13] and [18], is that of explicitly denoting time points and having tableau nodes labelled either by expressions intuitively stating that a formula holds in a given temporal interval, or "temporal constraints", i.e. linear inequalities on time points. Branch closure is reduced to unsatisfiability over the integers of the set of temporal constraints in the branch.

Let $C = \{start, finish, d_1, d_2, d_3, ...\}$ be a set of *time constants* (intuitively denoting time points, where *start* is the initial time point and *finish* the last one). A *state* is any expression of the form $c + n$, for $c \in C$ and $n \in \mathbb{Z}$. 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 a closed LTL formula in nnf (intuitively meaning that A holds at each point in the closed interval $[s, t]$). $[s, s]A$ will be abbreviated by $[s]A$.

Tableau nodes are labelled either by temporal constraints or labelled formulae (in this last case they are called *logical nodes*). If S is a finite set of formulae in nnf and K is the singleton $\{finish \leq start + k\}$ 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 $\{[start]A \mid A \in S\} \cup K$. Nodes are expanded by application of the rules in Table 1, where the expressions $s, t, s', t', ..$ are elements of Σ, C is a set of time constants and $P = \{a_1, a_2, a_3, ...\}$ is a set of "fresh" parameters (whose role is to "give names" to elements of the object domain), treated as individual constants. Rules for the equality predicate may be added to the rules in Table 1, in a standard way. 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.

In the δ-rule, c is a fresh element of C and a is a fresh element of P. In the γ-rule, u is any ground term occurring in the branch. Note that a sort of contraction is implicit in the β-rule and in the δ-rule: in both cases, 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. Moreover, logical formulae are in general not automatically "consumed" by expanding them; in particular, one may need to expand a universal formula several times by means of the γ-rule, as usual.

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. Similarly, the δ-rules can be read as stating that if $\exists x A(x)$ holds in the non empty interval $[s, t]$, then either $A(a)$ holds at each point of the interval for the same object a (leftmost branch), or $A(a)$ holds for the same object A in a first segment $[s, c]$ of the interval (possibly with $c = s$), and in the remaining part $\exists x A(x)$ holds again.

The interval and conflict resolution rules augment the set of temporal constraints in the tableau. 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 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 a 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.

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

Logical rules			
α-rule	**β-rule**	**$\mathcal{U}$-rule**	**$\Box$-rule**
$\dfrac{\begin{array}{c}[s,t]\,A\wedge B\\ s\le t\end{array}}{\begin{array}{c}[s,t]\,A\\ [s,t]\,B\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A\vee B\\ s\le t\end{array}}{[s,t]\,A \quad \begin{array}{c}s\le c\\ c\le t\\ [s,c-1]\,A\\ [c]\,B\\ [c+1,t]\,A\vee B\\ c\in C \text{ fresh}\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A\mathcal{U}B\\ s\le t\end{array}}{\begin{array}{c}[c]\,B\\ t+1\le c\\ [t+1,c-1]\,A\\ [s+1,t]\,A\vee B\\ c\in C\ \mathit{fresh}\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,\Box A\\ s\le t\end{array}}{[s+1,\mathit{finish}]\,A}$
γ-rule	**δ-rule**	**$\mathcal{S}$-rule**	**$\overleftarrow{\Box}$-rule**
$\dfrac{\begin{array}{c}[s,t]\,\forall x A(x)\\ s\le t\end{array}}{[s,t]\,A(u)}$	$\dfrac{\begin{array}{c}[s,t]\,\exists x A(x)\\ s\le t\end{array}}{[s,t]\,A(a) \quad \begin{array}{c}s\le c\\ c\le t\\ [s,c]\,A(a)\\ [c+1,t]\,\exists x A(x)\\ c\in C\ \mathit{fresh}\\ a\in P\ \mathit{fresh}\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,A\mathcal{S}B\\ s\le t\end{array}}{\begin{array}{c}[c]\,B\\ c\le s-1\\ [c+1,s-1]\,A\\ [s,t-1]\,A\vee B\\ c\in C\ \mathit{fresh}\end{array}}$	$\dfrac{\begin{array}{c}[s,t]\,\overleftarrow{\Box}A\\ s\le t\end{array}}{[\mathit{start},t-1]\,A}$

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

In the following, if $\mathcal{B}$ is a tableau branch, $const(\mathcal{B})$ denotes the set of temporal constants (elements of C) occurring in $\mathcal{B}$, 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 natural numbers. 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 $c\in C, n\in\mathbb{Z}$.*

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 $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$, then $\mathcal{M}_i \models A$.*
   (b) *$\mathcal{B}$ is* satisfiable *in $\mathcal{M}$ iff there exists a temporal mapping $\mathcal{I}$ for C such that $\langle\mathcal{M}, \mathcal{I}\rangle \models \mathcal{B}$.*

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

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

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

- *The interval rule is applied at most once to each interval.*
- *Every logical node is expanded at most once by means of a logical rule, but for nodes of the form $[s,t]\forall xA$, that can be expanded more than once.*
- *If a node $[s,t]\forall xA$ is expanded twice to, respectively, $[s,t]A(u)$ and $[s,t]A(u')$, via two applications of the γ-rule, then the ground terms u and u' are syntactically different.*
- *Each conflict resolution rule is applied at most once to each interval (second conflict resolution rule) or pair of intervals (first conflict resolution rule)*
- *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.*

Obviously, like in classical tableau systems, a canonical open branch may be infinite, because of the possible reapplication of the γ rule to a same node. More important, a priori, nothing ensures that a canonical *closed* branch is finite. In fact, an infinite branch may be closed (i.e. unsatisfiable) because its infinite set K of constraints is unsatisfiable, yet any finite subset of K has a (finite) solution. As an example, consider $K = \bigcup_{i \in \mathbb{N}} \{c_i \leq c_{i+1} - 1\}$: each finite subset of K has a solution, however K is not finitely satisfiable. Hence, in principle, a canonical tableau might be closed (all its branches being closed) and yet infinite (at least one branch being infinite). However, in the next section, we show that, thanks to the presence of the initial constraint $finish \leq start + k$ (setting a bound on the size of the searched models), closed tableaux are always finite.

5 Properties of the Bounded Tableau Calculus

In this section we establish the key properties of the calculus defined above. For space reasons, we do not give the full proofs here, which can however be found in [10]. The system can easily be proved to be sound:

Theorem 3 (Soundness w.r.t. k-Unsatisfiability). *Let S be a set of closed formulae. If there exists a closed tableau for $S \cup \{finish \leq start + k\}$ then S is k-unsatisfiable. A fortiori, if there exists a finite closed tableau for $S \cup \{finish \leq start + k\}$, then S is k-unsatisfiable.*

Conversely, any complete open branch describes some model of its initial formulae. Such a result needs some preliminary lemmas and definitions. First of all, the invertibility of all the logical rules with respect to their logical nodes can be established, in the following form:

Lemma 1. *1. Let $\mathcal{M} = \langle T, \sigma \rangle$ be a temporal interpretation and $\mathcal{I}$ a temporal mapping with range T. For every logical rule but the γ-rule*

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

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

2. Let $\mathcal{B}$ be an open and complete branch of a tableau and let $[s,t]\forall x A(x)$ be a node in $\mathcal{B}$, expanded via a γ-rule. Let $\mathcal{M}$ be a temporal interpretation such that its object domain is exactly the set of ground terms in the branch, and $\mathcal{M}(u) = u$ for any ground term u in $\mathcal{B}$. Suppose that, for any labelled node $[s,t]A(u)$, where u is a ground term, for each i in $[I^*(s), \mathcal{I}^*(t)]$, $\mathcal{M}_i \models A(u)$. Then, for any such i, $\mathcal{M}_i \models \forall x A(x)$.

The following definition captures the idea of a sequence of applications of the β-rule, each of them expanding an expansion of the previous one, and the similar notion for the case of the δ-rule.

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 δ-node in $\mathcal{B}$ is a node of the form $[s,t]\exists x A(x)$. 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 corresponding notions of δ-chain, k-length-δ-chain and root node of a δ-chain are defined similarly.

A tableau branch may contain an infinite number of β-chains or δ-chains. Moreover, in principle, each chain might be of unbounded length. However, if the branch is satisfiable by a model $\mathcal{M}$ whose temporal frame is $\langle 0, ..., k\rangle$, each chain has at most $k+1$ elements. This fact is stated by the next lemma, whose proof is similar to the corresponding one in [7].

Lemma 2. *If $\mathcal{B}$ is a canonical open branch and its constraints have a solution $\mathcal{I}$ whose range is $\langle 0, ..., n\rangle$, then $\mathcal{B}$ contains no k-length-β-chain or k-length-δ-chain with $k > n$.*

The previous lemmas enable us to prove the following result:

Lemma 3. *If $\mathcal{B}$ is a complete and open tableau branch and $\mathcal{I}$ is a solution of the set of 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. Suppose that $\mathcal{I}$ is a solution for the constraints in $\mathcal{B}$ and let $T = \langle 0, ..., n\rangle$ be the range of $\mathcal{I}$. Let $\mathcal{M}$ be $\langle T, D, \delta_P, \delta_F\rangle$ where D is the set of ground terms in $\mathcal{B}$; for each i in T, sequence $t_1, .., t_n \in D^n$, and n-ary predicate p: $t_1, ..., t_n \in \delta_P(p)$ iff there is a node $[s,t]p(t_1, ..., t_n)$ in $\mathcal{B}$ such that $\mathcal{I}^*(s) \leq i \leq \mathcal{I}^*(t)$; for any n-ary function symbol f and i in T: $\delta_F(f)(t_1, ..., t_n) = f(t_1, ..., t_n)$.

We need to prove that, for any labelled formula $[s,t]F \in \mathcal{B}$ such that $[I^*(s), I^*(t)] \neq \emptyset$, and for any $i \in [I^*(s), I^*(t)]$: $\mathcal{M}_i \models F$ (the case of an empty interval being trivial). In order to do that, we define an order $<$ on the labelled formulae in $\mathcal{B}$ as follows. Given $[s,t]F \in B$ and $[s',t']F' \in \mathcal{B}$, we set $[s,t]F < [s',t']F'$ iff either F is a strict subformula of F', or else $[s,t]F$ is obtained by applying either a β-rule or a δ-rule or a $\mathcal{S}$-rule or an $\mathcal{U}$-rule to $[s',t']F'$.

Note that the order $<$ is partial and all the labelled formulae $[s,t]L$ in $\mathcal{B}$ where L is a literal are minimal for it. *The order $<$ is well founded*; in order to prove it, the only delicate point is the second item in the definition of $<$. In fact, a priori, there might be an infinite decreasing $<$-chain, whose starting point is either a β-formula or a δ-formula or a $\mathcal{S}$-formula or a $\mathcal{U}$-formula. However, since $\mathcal{I}$ is a solution to the constraints in $\mathcal{B}$ (with at most $n+1$ time points), Lemma 2 rules out such a possibility.

Thus, it suffices to prove by induction on $<$, that if $[s,t]F \in \mathcal{B}$ then, for any $i \in [\mathcal{I}^*(s), \mathcal{I}^*(t)]$, $\mathcal{M}_i \models F$. The inductive step uses Lemma 1.

The proof may be slightly modified so as to hold also in the case where equality rules are present. Of course, in order to define the domain of the model $\mathcal{M}$, the set of ground terms needs to be quotiented w.r.t. equality. ■

As a consequence, we have the following result:

Theorem 4 (Quasi-Completeness w.r.t.k-unsatisfiability). *Let S be a set of formulae and k a natural number. If S is k-unsatisfiable, then there exists a closed tableau for $S \cup \{finish \leq start + k\}$*

The reason why we call such a result *quasi-completeness*, rather than completeness, is the following. As we have already remarked, a priori a closed branch might be infinite, because, in principle, we could have an infinite set K of temporal constraints in the branch, such that any finite subset of K has a (finite) solution but K itself has no solution. Thus, in principle, we could have an infinite tableau for a formula A that is indeed closed without being able to finitely recognize this fact. Such an infinite closed tableau would not be a "refutation" since "being a refutation" is meant to be a recursive predicate. Thus, if infinite closed tableaux might exists, Theorem 4 would not imply that the tableau calculus is an effective complete proof system, enabling us to semidecide the k-validity problem for $FO\text{-}LTL^{fin}$.

However, thanks to the presence of the initial constraint $finish \leq start + k$, any closed branch is indeed finite. In fact, if a branch contains an infinite set of constraints K and K has no solution, some finite subset of K has no solution. This property holds because:

1. Since the constraint $finish \leq start + k$ belongs to K, and, for each temporal constant c, the constraint $c \leq finish$ (implicitly) belongs to K, any solution for K is necessarily upper bounded by k.
2. If each finite subset of a denumerable set of constraints K has a solution that is upper bounded by k, then K itself has a solution upper bounded by k.

The second fact is a corollary of a more general property:
Let K be a denumerable set of constraints and k a function from the set of constants C occurring in K to the natural numbers. If any finite subset K' of K has a solution $\mathcal{I}$ such that, for any constant $c \in K'$ $\mathcal{I}(c) \leq k(c)$, then K has a solution $\mathcal{I}^+$ such that, for each constant $c \in K$, $\mathcal{I}^+(c) \leq k(c)$.

Proof. This property can be proved by standard combinatorial techniques. Let $c_1, c_2, ...$ be an enumeration of the set C of constants appearing in K. Let $e_1, e_2, ...$ be an enumeration of the elements of K; such an enumeration induces an enumeration $K_1, K_2, ...$ of finite subsets of K, where $K_i = \{e_1, ..., e_i\}$. Say that the finite set of constants appearing in S_i is C_i. For each $i \geq 1$, by hypothesis there is some solution $\mathcal{I}_i$ of K_i such that $\mathcal{I}(c) \leq k(c)$ for any constant $c \in C_i$; let us extend such a solution $\mathcal{I}_i$ to a function $\mathcal{I}'_i$ from C to the natural numbers, by assigning any value to the constants in $C - C_i$. This operation can be visualized like the construction of a table, possibly having an infinite number of lines and an infinite numbers of columns:

	c_1	c_2	c_3	$\cdots$
K_1	$\mathcal{I}'_1(c_1)$	$\mathcal{I}'_1(c_2)$	$\mathcal{I}'_1(c_3)$	$\cdots$
K_2	$\mathcal{I}'_2(c_1)$	$\mathcal{I}'_2(c_2)$	$\mathcal{I}'_2(c_3)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

where the labels of the columns are the constants in C and the labels of the lines are the finite subsets K_i. Any line labelled by K_i corresponds to the codomain of the function $\mathcal{I}'_i$ and, for each j such that $c_j \in C_i$, the case (i,j) actually contains the value $\mathcal{I}_i(c_j)$, where $\mathcal{I}_i$ is a "local" solution for K_i. Given any j, let us call "OK cases" of the column j those cases (i,j) in the column such that (i,j) contains $\mathcal{I}_i(c_j)$. In order to obtain the required global solution $\mathcal{I}^+$, its values $\mathcal{I}^+(j)$ are defined by induction on j, as follows.
For $j = 1$, since the values appearing in all the OK cases of the first column are bounded by $k(c_1)$, and there are infinitely many such OK cases, there is at least a number n_1 which appears infinitely often in these cases. Set $\mathcal{I}^+(c_1) = n_1$, then update the table by erasing each line i such that the content of the case $(i,1) \neq n_1$. (Note that an infinite numbers of lines is left).
For $j > 1$, the table updated at the previous stage is considered. Again, there is at least a number n_j which appears in all the infinitely many OK cases of the j-th column. Set $\mathcal{I}^+(c_j) = n_j$, then "erase" each line i such that the content of the case $(i,j) \neq n_j$. By construction, $\mathcal{I}^+$ satisfies each constraint in C, and, for any $c \in C$, $I^+(c) \leq k(c)$. ∎

Hence, as a consequence of Theorems 3 and 4 we obtain:

Theorem 5. *Let S be a set of FO-LTL^{fin} formulae, k be any positive integer and $K = \{finish \leq start + k\}$.*

1. *If there exists a closed tableau for $S \cup K$, such a tableau is necessarily finite and S is k-unsatisfiable. (Soundness with respect to k-unsatisfiability)*
2. *If S is k-unsatisfiable, then there exists a finite closed tableau for $S \cup K$. (Completeness with respect to k-unsatisfiability)*

The tableau system defined in this work is not sound w.r.t. validity *tout court*: obviously, a formula A might be true in all the temporal interpretations whose time structure is bounded by a given number k without being valid. However, we can slightly modify the tableau definition, simply removing the initial constraint $finish \leq start + k$, to obtain a calculus that is indeed sound w.r.t. validity: if a closed tableau for $\neg A$ exists, then $\neg A$ is unsatisfiable, i.e. A is valid.

However, in the unbounded calculus, a closed tableau for S may contain an infinite branch. For instance, take S to be the unsatisfiable set of formulae $\{\Box\exists x p(x)),\ \Diamond\forall y \neg p(y)\}$. Because of the implicit contraction in the δ-rule, the rightmost branch of any tableau for S will contain an infinite set of constraints of the form $\{c_1 + 1 \leq c_2 \leq finish, c_2 + 1 \leq c_3 \leq finish, ...\}$: the absence of the initial constraint on the size of the searched model enables us to continue the construction of a δ-chain *ad libitum*. Each finite subset of such a set of constraints has a solution, but the set itself has no (finite) solution at all. Thus, although the tableau for S is closed, such a tableau is not a *refutation* of S.

Hence the unbounded calculus is *incomplete* w.r.t. to validity. Note that the analogous of Theorem 4 holds for the unbounded calculus: given any set S of unsatisfiable formulae, a closed tableau for S does exist. Yet, this does not provide a semidecision procedure for validity.

6 Conclusions

The contribution of this paper is twofold: we report a negative result on $FO\text{-}LTL^{fin}$, i.e. first order linear temporal logic over finite temporal frames, and prove a positive one. The negative result shows the impossibility of defining an effective proof system that is complete with respect to $FO\text{-}LTL^{fin}$ validity. The analogous result for "standard" LTL equipped with *future* time operators had been proven in [1] and [16]; however, such proofs are not immediately exploitable in the case of LTL^{fin}, since they essentially depend on the possibility of encoding arithmetical formulae in LTL. As a byproduct of our negative result for LTL^{fin}, first order LTL-validity of formulae containing only *past* time operators is not recursively enumerable either. As far as $FO\text{-}LTL^{fin}$ with the $\Box$ operator only is concerned, it would be interesting to check whether a result analogous to [15] holds (at a first check, the answer seems to be positive, but the details of the proofs are still to be worked out).

On the positive side, we have defined a labelled tableau calculus which is sound and complete w.r.t. bounded validity, that can be the basis for the construction of theorem provers for applications where information is needed about the validity of a temporal formula in all models where the number of time points does not exceed a given bound. This is the case, for instance, of dynamic constraint management in databases, where the question of the "safety" of a given sequence of updates - of a fixed length - with respect to a given constraint is important, as well as of the search of bounded length plans. The tableau calculus we propose makes use of annotations on formulae, indicating the time intervals in which formulae are taken to hold, hence it is formulated in a *labelled deduction* style, similarly to [13, 18]. In the case of linear temporal logic, the use of labels enables us to encode, inside the proof system itself, the information that time points "behave" as natural numbers. With respect to non labelled calculi (see for example [19]), such an approach presents, in principle, the following main advantages: the possibility of "immediately executing" *eventualities* (i.e. formulae with an existential commitment), without the need to systematically choose the specific point where the considered formula must hold, and a symmetric treatment of past and future time operators, without the need of a preliminary transformation of formulae into a "separated" form (see [12] and [4]).

Acknowledgments. The authors wish to thank Laurent Rosaz for useful discussions and suggestions.

References

1. M. Abadi. The power of temporal proofs. *Theoretical Computer Science*, 65:35–83, 1989.
2. M. Abadi and Z. Manna. Nonclausal deduction in first-order temporal logic. *Journal of the Association for Computing Machinery*, 37:279–317, 1990.
3. S. Abiteboul, L. Herr, and J. Van den Bussche. Temporal versus first-order logic to query temporal databases. In *Proc. of the 15th Int. Conf. on Principles of Databases (PODS'96)*, 1996.

4. 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.
5. V. Benzaken, S. Cerrito, and S. Praud. Vérification statique de contraintes d'intégrité dynamiques. In *Proc. of 15èmes Journées Bases de Données Avancées (BDA99)*, Bordeaux, France, October 1999. To appear.
6. N Bidoit and S. De Amo. A first step towards implementing dynamic algebraic dependencies. *Theoretical Computer Science*, 2(190):115–149, january 1998.
7. S. Cerrito and M. Cialdea Mayer. Bounded model search in linear temporal logic and its application to planning. In H. De Swart, editor, *International Conference on Automated Reasoning with Analytical Tableaux and Related Methods*, pages 124–140. Springer, 1998.
8. S. Cerrito and M. Cialdea Mayer. Using linear temporal logic to model and solve planning problems. In F. Giunghiglia, editor, *Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, pages 141–152. Springer Verlag, 1998.
9. S. Cerrito, M. Cialdea Mayer, and S. Praud. First order linear temporal logic over finite time structures is not semi-decidable. Tecnical Report LRI n. 1208, available at `http://www.dia.uniroma3.it/~cialdea/papers/nonre.ps`. Presented at the Workshop Methods for Modalities 1 (M4M) (Amsterdam, May 1999).
10. S. Cerrito, M. Cialdea Mayer, and S. Praud. A tableau calculus for first order linear temporal logic over bounded time structures. Technical Report LRI n. 1207, available at `http://www.dia.uniroma3.it/~cialdea/papers/foltl.ps`.
11. J. Chomicki. Temporal query languages: a survey. In D.M. Gabbay and H.J. Ohlbach, editors, *Temporal Logic: ICTL'94*, volume 827 of *Lecture Notes in Computer Science*, pages 506–534. Springer-Verlag, 1994.
12. D. Gabbay. Declarative past and imperative future: Executable temporal logic for interactive systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proc. of Colloquium on Temporal Logic in Specification*, number 398 in LNCS, pages 409–448. Sprinber-Verlag, 1989.
13. 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.
14. D. Harel. Recurring dominoes: Making the higly undecidable higly understandable. *Annals of Discrete Mathematics*, 24:51–72, 1985.
15. M Kaminski and C.K. Wong. The power of the "always" operator in first-order temporal logic. *Theoretical Computer Science*, pages 271–281, 1996.
16. F. Kröger. On the interpretability of arithmetic in temporal logic. *Theoretical Computer Science*, 73:47–61, 1990.
17. P.H. Schmitt and J. Goubault-Larrecq. A tableau system for full linear temporal logic. Draft, available at: http://www.dyade.fr/fr/actions/vip/jgl/ltl2.ps.gz.
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. P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28:119–152, 1985.