A Tableau System for Linear Temporal Logic

Peter H. Schmitt Jean Goubault-Larrecq*†
P.H.Schmitt@ira.uka.de Jean.Goubault@ira.uka.de
Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe, D-76128 Karlsruhe

October 16, 1996

Abstract

We present a sound, complete and terminating tableau system for the propositional logic of linear time with only $\Box$ and $\Diamond$ as temporal operators.

1 Introduction

Deduction calculi for propositional linear temporal logic and their realisation in theorem proving programs play an important part in verification and analysis of systems. We are concerned here with one particular class of calculi, the tableau calculi, which seem to be most popular in this area and also well adapted to the task. As a first introduction to this method we refer to the overview paper [Wol85]. A more recent tableau calculus is treated at length in the book [MP95] and has also been implemented in the STeP System: see [MAB+94] and [BBC+96].

One drawback of all calculi for linear temporal logic proposed so far, as compared to calculi in classical propositional logic, is the fact that termination of the procedure cannot be determined locally. In [Wol85] e.g. a loop-check has to be performed during the expansion of the tableau to detect recurring node labels. The tableau system in [MP95] depends crucially on the global concept of maximal strongly connected subgraphs of the tableau. Both approaches necessitate the use of involved data structures and algorithms in an implementation, which limits the efficiency of the resulting theorem proving system. In this paper, we present a calculus that terminates when no further expansion rule is applicable, which is guaranteed to happen. The relevant data structure basically consists of the current branch, which is simply a set of temporal formulas. The formulas we consider in this paper only use $\Box$ and $\Diamond$ as temporal operators, and the next-time operator $\bigcirc$ is not included. This fragment, let us call it LTL_0, is less expressive than full linear time logic, LTL. An extension of the calculus to full LTL is in preparation. But the restricted temporal language also deserves attention, if only for the reason that the satisfiability problem for LTL_0 is merely NP-complete [SC85], while it is PSPACE-complete for LTL. It may therefore be advantageous to submit problems that happen to belong to the fragment LTL_0 to a special proof procedure, rather than to feed them into a prover that treats all of LTL.

Another approach to tableau calculi for linear temporal logic was taken in [HI94] and continued in [Gei95]. In these papers the logical calculus is augmented by introducing integer constraints, and the use of algorithms from integer programming is suggested. The present paper has profited from this work, in that we use similar signed formulas (sometimes also called labelled formulas) in our calculus.

*Research funded by the HCM grant 7532.7-06 from the European Union.
†On leave from Bull Corporate Research Center, rue Jean Jaurès, F-78340 Les Clayes sous Bois.

2 Linear Temporal Logic

The formulas of the linear temporal logic LTL_0, which we shall denote by capital letters in the range F, G, H, ..., are built from propositional variables A, B, C, ... by the following grammar:

$$F ::= \quad A \mid \neg F \mid F \wedge F \mid F \vee F \mid \Box F \mid \Diamond F$$

$F \rightarrow G$ is taken to be an abbreviation for $\neg F \vee G$, and $F \leftrightarrow G$ for $(F \rightarrow G) \wedge (G \rightarrow F)$.

We call *literal* L any formula of the form A or $\neg A$, where A is a variable.

The semantics is as follows. A *structure* S is a pair (s, ξ) where $s = (s_0, s_1, \ldots)$ is an ω-sequence of states in which all the states are distinct and ξ is a valuation mapping each s_i, $i \in \mathbb{N}$, to some set of propositional variables (the variables that are true in state s_i). We may identify states with natural integers. An *interpretation* (S, i) is a pair of a structure S and a state i. Then, we say that a formula F holds under (S, i), and we write $S, i \models F$, in the following cases:

- $S, i \models A$ iff $A \in \xi(s_i)$;
- $S, i \models \neg F$ iff $S, i \not\models F$;
- $S, i \models F \wedge G$ iff $S, i \models F$ and $S, i \models G$;
- $S, i \models F \vee G$ iff $S, i \models F$ or $S, i \models G$;
- $S, i \models \Box F$ iff for every $j \geq i$, $S, j \models F$;
- $S, i \models \Diamond F$ iff for some $j \geq i$, $S, j \models F$;

A formula is said to be *valid in a structure* S, denoted by $S \models F$, if $S, i \models F$ holds for all i. A formula is *valid* if it valid in every structure.

The tableau calculus to be described in the next section will operate not just on formulas but will also exploit additional "semantic" information in the form of signed clauses and constraints. As a preparation we introduce the notion of time points. We shall make use of an infinite vocabulary T_c of *time constants* c_0, c_1, ..., which are simply fresh names. *Time points* s, t, ... are defined as pairs (c, n), where c is a time constant and $n \in \mathbb{Z}$. We define $(c, n) + m$ as $(c, n + m)$ for any $m \in \mathbb{Z}$, and similarly $(c, n) - m = (c, n - m)$. We abbreviate $(c, 0)$ as c, (c, n) as $c + n$ and $(c, -n)$ as $c - n$, as there won't be any risk of confusion. The set of all time points will be denoted by T. Intuitively, time constants and time points denote states in a structure.

We define *intervals* I as expressions of the form $[s, t]$ or $[s, +\infty[$, where s and t are time points. Intuitively, $[s, t]$ denotes the set of all states between s and t inclusively, and $[s, +\infty[$ denotes the set of all states at or above s. Note that $[s, t]$ denotes the empty set of states whenever $s \geq t + 1$. We abbreviate $[s, s]$ as $[s]$.

A *clause* is a multiset C of formulas. Notice that this generalizes the usual definition of a clause, which would require that all formulas in C be literals. A *signed clause* is either $I\ C$ or $|\infty|\ C$, where C is a clause and I is an interval. Will will use capital Greek letters like Φ, Ψ to denote signed clauses and small Greek letters like σ to denote intervals or the symbol $|\infty|$. To explain the semantics of signed clauses we consider *time assignments* τ. These are mappings from T_c into $\mathbb{N}$ satisfying $\tau(c_0) = 0$. Extend τ to a mapping $T \rightarrow \mathbb{Z}$ by letting $\tau(c + n) = \tau(c) + n$. Given a structure S and a time assignment τ, we define:

- $S, \tau \models [s, t]C$ iff for every i such that $\tau(s) \leq i \leq \tau(t)$ $S, i \models F$ for some formula F in C,
- $S, \tau \models [s, +\infty[C$ iff for every i such that $\tau(s) \leq i$, $S, i \models F$ for some formula F in C,
- $S, \tau \models |\infty|\,C$ iff for infinitely many i, $S, i \models F$ for every formula F in C.

Notice the asymmetry in this definition: a clause with prefixed interval is treated as a logical disjunction, while for $|\infty|\,C$ the multiset C is considered as a logical conjunction.

Expressions of the form $s \leq t$, with s and t time points are called *constraints*. A *constraint set* K is any finite set of constraints. We say that a time assignment τ *satisfies* a constraint $s \leq t$, denoted by $\tau \models s \leq t$, if and only if $\tau(c) \leq \tau(c') + n$; and τ satisfies a constraint set K if and only if it satisfies all the constraints in K. K is *satisfiable* if it is satisfied by some time assignment.

$$\frac{[s,t]C, F \land G}{\begin{array}{c}[s,t]C, F\\ [s,t]C, G\end{array}} \qquad \frac{[s,t]C, \Box F}{[s,t]C \;\Big|\; \begin{array}{c}[s,u-1]C\\ [u,+\infty[F\\ s \le u \le t\\ u \text{ fresh}\end{array}} \qquad \frac{[s,t]C, \Diamond F}{[s,t]C \;\Big|\; \begin{array}{c}[u]F\\ [u+1,t]C\\ s \le u\\ u \text{ fresh}\end{array}}$$

$$\frac{[s,t]C, F \lor G}{[s,t]C, F, G}$$

$$\frac{[s,t]C, \neg(F \land G)}{[s,t]C, \neg F, \neg G} \qquad \frac{[s,t]C, \neg\Box F}{[s,t]C \;\Big|\; \begin{array}{c}[u]\neg F\\ [u+1,t]C\\ s \le u\\ u \text{ fresh}\end{array}} \qquad \frac{[s,t]C, \neg\Diamond F}{[s,t]C \;\Big|\; \begin{array}{c}[s,u-1]C\\ [u,+\infty[\neg F\\ s \le u \le t\\ u \text{ fresh}\end{array}}$$

$$\frac{[s,t]C, \neg(F \lor G)}{\begin{array}{c}[s,t]C, \neg F\\ [s,t]C, \neg G\end{array}}$$

$$\frac{[s,t]C, \neg\neg F}{[s,t]C, F}$$

Figure 1: Logical rules, closed intervals

3 A Tableau System

A *tableau* $\mathcal{T}$ is a set of branches, where a branch (B, K) is a pair formed of a set B of signed clauses and a constraint set K. To prove that a formula F is valid, we set up the initial tableau $\mathcal{T}_0$, consisting of a single branch (B_0, K_0), where B_0 has $[c_0]\neg F$ as unique element, and K_0 is empty. The tableau procedure proceeds by applying two different kinds of steps, *expansions steps* and *closures*. An expansion step on a tableau $\mathcal{T}$ is performed by choosing a branch (B, K) in $\mathcal{T}$ and an unused signed clause Φ in B, and by applying the matching logical *tableau rule* as given in Figures 1, 2, and 3. Application of a tableau rule consists in marking the premise clause as used and in extending the branch (B, K) by adding the signed clauses and constraints in the conclusion of the tableau rule to B and K respectively. If the conclusions of a rule consists of two or three alternatives, which is denoted in the figures by a vertical bar, then (B, K) is split in two resp. three extensions. In each rule, u denotes a fresh time constant, that is, one that does not appear already on the branch. We also use C, F throughout as an abbreviation for the clause $C \cup \{F\}$.

Observe that the logical rules on open intervals are mostly the same as those on closed intervals (the conditions $u \le t$ are dropped in $\Diamond$ and $\neg\Box$ rules, and an $|\infty|$ case is added).

We say that a signed clause $\sigma\ C$ is *atomic* whenever C consists of literals only. A branch (B, K) is atomic when all its unused signed clauses are atomic. It is obviously not possible to apply an extension step to an atomic branch.

Extending the initial tableau $(\{[c_0]\neg F\}, \emptyset)$ may be viewed as a systematic search for a counter-model of F. Each branch may then be looked at as a partial counter-model. It will be shown below that if a tableau is reached with all its branches unsatisfiable, then $[c_0]\neg F$ is also unsatisfiable and thus F is valid.

A branch (B, K) is *satisfiable* if there is a structure S and a time assignment τ such that $S, \tau \models C$ for each signed clause C in B and $\tau(s) \le \tau(t)$ for all constraints $s \le t$ in K.

Detecting the unsatisfiability of a branch is the objective of the closure steps. We postpone the details of closing branches until Section 5. For now, we just assume that there is a procedure *closing* that after a finite number of steps marks an atomic branch as closed iff it is unsatisfiable.

4 Tableau Expansion

Let $T_1 \subseteq T_2$ be subsets of the set of all time constants T_c and τ a time assignment $T_1 \to \mathbb{N}$. We say that a time assignment τ_2 *extends* τ to T_2 if τ_2 agrees with τ on T_1.

The basic step in proving soundness and completeness of the expansion steps is the following Theorem.

$$\frac{[s,+\infty[C, F \wedge G}{\begin{array}{c}[s,+\infty[C, F\\ {[s,+\infty[C, G}\end{array}}$$

$$\frac{[s,+\infty[C, F \vee G}{[s,+\infty[C, F, G}$$

$$\frac{[s,+\infty[C, \neg(F \wedge G)}{[s,+\infty[C, \neg F, \neg G}$$

$$\frac{[s,+\infty[C, \neg(F \vee G)}{\begin{array}{c}[s,+\infty[C, \neg F\\ {[s,+\infty[C, \neg G}\end{array}}$$

$$\frac{[s,+\infty[C, \neg\neg F}{[s,+\infty[C, F}$$

$$\frac{[s,+\infty[C, \Box F}{\begin{array}{c|c}[s,+\infty[C & [s,u-1]C\\ & {[u,+\infty[F}\\ & s \leq u\\ & u \text{ fresh}\end{array}}$$

$$\frac{[s,+\infty[C, \Diamond F}{\begin{array}{c|c|c}[s,+\infty[C & [u]F & |\infty|\,F\\ & {[u+1,+\infty[C} & \\ & s \leq u & \\ & u \text{ fresh} & \end{array}}$$

$$\frac{[s,+\infty[C, \neg\Box F}{\begin{array}{c|c|c}[s,+\infty[C & [u]\neg F & |\infty|\,\neg F\\ & {[u+1,+\infty[C} & \\ & s \leq u & \\ & u \text{ fresh} & \end{array}}$$

$$\frac{[s,+\infty[C, \neg\Diamond F}{\begin{array}{c|c}[s,+\infty[C & [s,u-1]C\\ & {[u,+\infty[\neg F}\\ & s \leq u\\ & u \text{ fresh}\end{array}}$$

Figure 2: Logical rules, open intervals

Theorem 1 *For every rule in Figures 1, 2, and 3, for every structure S, for every time assignment τ on a set T containing all time constants occurring in the premise Φ of the rule, $S, \tau \models \Phi$ if and only if there is a conclusion of the rule and a time assignment τ' extending τ such that (1) τ' satisfies the constraints of the conclusion and (2) $S, \tau' \models \Phi'$ for every signed clause Φ' in the conclusion.*

Proof: Figure 1: this is obvious for the rules not involving the modal connectives. Consider the rule with premise $[s,t]C, \Box F$. Then $S, \tau \models [s,t]C, \Box F$ iff $S \models [i,j]C, \Box F$, where i, resp. j, is the value of s, resp. t, under τ. If $S \models [i,j]C, \Box F$, then we distinguish two cases. In the first case, C holds on the whole interval $[i,j]$: then the left-hand conclusion holds under time assignment τ. In the second case, there is a state k, $i \leq k \leq j$, where C does not hold. Then, $\Box F$ must hold in state k, i.e. $S, k' \models F$ for every $k' \geq k$, i.e. $S \models [k,+\infty[F$. Moreover, we may choose k as the *earliest* state where C does not hold, so $S \models [i,k-1]C$. Take τ' extending τ so that $\tau'(u) = k$: τ' satisfies the constraint $s \leq u \leq t$, and every signed clause of the right-hand conclusion holds under S, τ'.

Conversely, if τ' extends τ so that $S, \tau' \models [s,t]C$, then $S, \tau \models [s,t]C$ and trivially $S, \tau \models [s,t]C, \Box F$. And if τ' extends τ so that (1) and (2) hold on the right-hand conclusion, then letting i, j, k be the values of s, t, u under τ' (also under τ for the first two), we have $S \models [i,k-1]C$, $S \models [k,+\infty[F$, and $i \leq k \leq j$. In particular, we have $S \models [i,k-1]C$ and $S \models [k,j]\Box F$, so $S \models [i,j]C, \Box F$, so the premise holds under τ.

$$\frac{|\infty|\,C, F \wedge G}{|\infty|\,C, F, G} \qquad \frac{|\infty|\,C, F \vee G}{|\infty|\,C, F \;\Big|\; |\infty|\,C, G} \qquad \frac{|\infty|\,C, \Box F}{\begin{array}{c}|\infty|\,C\\ {[u,+\infty[F}\\ c_0 \leq u\end{array}} \qquad \frac{|\infty|\,C, \Diamond F}{\begin{array}{c}|\infty|\,C\\ |\infty|\,F\end{array}} \qquad \frac{|\infty|\,C, \neg\neg F}{|\infty|\,C, F}$$

$$\frac{|\infty|\,C, \neg(F \wedge G)}{|\infty|\,C, \neg F \;\Big|\; |\infty|\,C, \neg G} \qquad \frac{|\infty|\,C, \neg(F \vee G)}{|\infty|\,C, \neg F, \neg G} \qquad \frac{|\infty|\,C, \neg\Box F}{\begin{array}{c}|\infty|\,C\\ |\infty|\,\neg F\end{array}} \qquad \frac{|\infty|\,C, \neg\Diamond F}{\begin{array}{c}|\infty|\,C\\ {[u,+\infty[\neg F}\\ c_0 \leq u\end{array}}$$

all u fresh

Figure 3: Logical rules, infinitely occurring events

Consider now the rule with premise $[s,t]C, \Diamond F$, and let's go a bit faster. If $[i,j]C, \Diamond F$ holds, then either $[i,j]C$ holds (left conclusion), or there is a state k in $[i,j]$ where C does not hold. In the latter case, choose k the *latest* such k. Then $[k+1,j]C$ holds, and $\Diamond F$ holds in state k. In particular, there is a state $k' \geq k$ such that F holds in state k'. Moreover, since $[k+1,j]C$ holds and $k' \geq k$, in particular $[k'+1,j]C$ holds. And $i \leq k'$ also holds, so (1) and (2) are verified on the right-hand conclusion if we take τ' extending τ by mapping u to k' (not k).

Conversely, if $[i,j]C$ holds (left conclusion), then $[i,j]C, \Diamond F$ holds too (premise). And if $[k',k']F$ and $[k'+1,j]C$ hold with $i \leq k'$ (right conclusion), then $[i,k']\Diamond F$ on the one hand and $[k'+1,j]C$ on the other hand, so $[i,j]C, \Diamond F$.

The cases of $[s,t]C, \neg\Box F$ and $[s,t]C, \neg\Diamond F$ are similar.

Figure 2. The cases are similar, except that in the $[s,+\infty[C, \Diamond F$ case, where we assume that $[i,+\infty[C, \Diamond F$, there might not be any latest $k \geq i$ such that C does not hold. This justifies the third possible conclusion: in that case, there is an infinite sequence $k_0 < k_1 < \ldots$ of states such that $k_0 \geq i$ and C does not hold at k_0, or k_1, or $\ldots$; therefore $\Diamond F$ must hold at k_0, and at k_1, and $\ldots$; so there are states $k'_0 \geq k_0$, $k'_1 \geq k_1$, $\ldots$, where F holds. This sequence k'_0, k'_1, $\ldots$, must be infinite, otherwise it would be bounded from above, hence k_0, k_1, $\ldots$, would also be bounded from above, which is impossible. So there is an infinitely increasing subsequence of states $k'_{\sigma_0} < k'_{\sigma_1} < \ldots$ at which F holds, i.e. $|\infty|F$ holds. Conversely, if $|\infty|F$ holds, there is a sequence $k_0 < k_1 < \ldots$ of states at which F holds; then for every $j \geq i$, there exists a k_p such that $k_p \geq j$, at which F holds, so $[i,+\infty[\Diamond F$ holds, and therefore $[i,+\infty[C, \Diamond F$ holds, i.e. the premise holds. The case of the rule of premise $[s,+\infty[C, \neg\Box F$ is similar.

Figure 3. The case of the rule with premise $|\infty|C, F \wedge G$ is obvious (remember that clauses under the $|\infty|$ sign are taken conjunctively).

As regards $|\infty|C, F \vee G$, it holds if and only if there is a sequence k of states $k_0 < k_1 < \ldots$ at which C and $F \vee G$ hold. Consider the subsequence k' of states at which C and F hold, and the subsequence k'' of states at which C and G hold. Every state in k belongs to k' or to k'', so one of k' and k'' must be infinite. If k' is infinite, then the left-hand conclusion $|\infty|C, F$ holds, and if k'' is infinite, then the right-hand conclusion holds. Conversely, if $|\infty|C, F$ or $|\infty|C, G$ holds, then clearly $|\infty|C, F \vee G$ too.

Look at the rule with premise $|\infty|C, \Box F$. If it holds, then there is a sequence $k_0 < k_1 < \ldots$ of states at which both C and $\Box F$ hold. In particular, $|\infty|C$ and $[k_0,+\infty[F$ hold: we let τ' extend τ by mapping u to k_0, then the conclusion holds. Conversely, if $|\infty|C$ and $[k,+\infty[F$ both hold, then let $k_0 < k_1 < \ldots$ be the sequence of states at which C holds. Then the subsequence of these states that are $\geq k$ is an infinite sequence of states at which C and $\Box F$ both hold, so the premise holds.

Consider now the rule with premise $|\infty|C, \Diamond F$. If it holds, then there is a sequence $k_0 < k_1 < \ldots$ of states at which both C and $\Diamond F$ hold. So on the one hand $|\infty|C$ holds, and on the other hand there are states $k'_0 \geq k_0$, $k'_1 \geq k_1$, $\ldots$, at which F holds. If k'_0, k'_1, $\ldots$, were bounded from above, so would be k_0, k_1, $\ldots$; so we can extract an infinite increasing subsequence $k'_{\sigma_0} < k'_{\sigma_1} < \ldots$ of states at which F holds. That is, $|\infty|F$ holds. Conversely, if both $|\infty|C$ and $|\infty|F$ hold, then let $k_0 < k_1 < \ldots$ be an infinite sequence of states at which C holds. At each k_p, C holds and there is also a later state at which F holds, so $\Diamond F$ holds at k_p. Therefore the premise $|\infty|C, \Diamond F$ holds.

The other rules of Figure 3 are similar.

□

Theorem 2 *Applying the rules in Figures 1, 2, 3 terminates.*

Proof: Define the following complexity measure: $|A| = 1$ for variables A, $|F \wedge G| = |F \vee G| = |F| + |G| + 2$, $|\neg F| = |\Box F| = |\Diamond F| = |F|+1$; $|[s,t]F_1, \ldots, F_n| = |[s,+\infty[F_1, \ldots, F_n| = ||\infty|F_1, \ldots, F_n| = |F_1|+\ldots+|F_n|$. If B is a branch of a tableau, then $|B|$ is the natural ordinal sum of the $\omega^{|\Phi|}$ where Φ ranges over the signed clauses on B (i.e., we compare branches by a multiset path ordering). Then every conclusion of a rule is smaller that its premise, so that branches decrease by application of each rule. □

Assume that a procedure *closing* is given that takes as input atomic tableau branches (B,K) and returns the answer *closed* iff (B,K) is not satisfiable.
We call a tableau $\mathcal{T}$ *closed* when all it branches are atomic and the procedure *closing* returns *closed* for all branches in $\mathcal{T}$.

Theorem 3 *A formula F from LTL_0 is valid iff from the initial tableau $\{([c_0]\neg F, \emptyset)\}$ a closed tableau can be reached by successive applications of expansion steps.*

Proof: Given a structure S and a time assignment τ we say that S, τ satisfies a tableau $\mathcal{T}$ if there is one branch (B, K) in $\mathcal{T}$ with $S, \tau \models (B, K)$.
Let $\mathcal{T}_0, \ldots \mathcal{T}_k$ be a sequence of tableaux, with $\mathcal{T}_0$ the initial tableau for F, $\mathcal{T}_{i+1}$ is obtained from $\mathcal{T}_i$ by one extension step for all $0 \leq i < k$ and $\mathcal{T}_k$ is closed. We claim that F is valid. By definition and the assumptions on the procedure *closing* $\mathcal{T}_k$ is not satisfiable. Using Theorem 1 repeatedly we conclude that $\mathcal{T}_0$ is not satisfiable, which is just another way of saying that F is valid.
Assume on the other hand that F is valid, thus the initial tableau $\mathcal{T}_0$ is not satisfiable. By Theorem 2, we know that by a finite number of expansion steps we will reach an atomic tableau $\mathcal{T}_k$. Repeated application of Theorem 1, now used in the reverse direction, shows that $\mathcal{T}_k$ is not satisfiable. By the assumption on *closing* this implies that $\mathcal{T}_k$ is indeed closed. □

Observe also that all rules are invertible, so that F is in fact valid iff a closed tableau can be reached by *any* maximal sequence of expansion steps: we don't have to backtrack on the choice of the expansion rule.

5 Closing Branches

Any procedure for closing atomic branches that satisfies the requirement set forth above Theorem 3 can be used together with the expansion rules to yield a complete and sound method to prove validity of formulas in LTL_0. In the next subsection we propose a procedure based on the propositional resolution calculus. Refinements to this procedure will be discussed in Subsection 5.2.

5.1 Closing by Resolution

The calculus we consider here is again a tableau calculus as described in Section 3, now using the rules of Figure 4 and 5. Unlike the previously considered rules those in Figure 5 need two premises in order to be applicable to a branch. Also, in contrast to the previous tableau rules and the closing rules, the resolution rules do not mark their premises as "used". Thus the same formula may be used in several resolution steps. However, it is never necessary to resolve on the *same pair* of signed clauses twice on the same branch.
The rules in Figure 5 basically amount to resolution with side-conditions on the way that the end-points of intervals are ordered.

Theorem 4 *The closing and resolution rules are sound, i.e. for every structure S, for every time assignment τ, if the premises of any resolution rule hold under S, τ, then so does one of its conclusions.*

Proof: Figure 4:
$[i, j]\varepsilon$ holds if and only if for every $i \leq k \leq j$, $S, k \models \varepsilon$, i.e. false. That is, $[i, j]\varepsilon$ if and only if there is no k such that $i \leq k \leq j$, namely if and only if $i \geq j + 1$: this justifies the leftmost topmost rule.

Similarly, $[s, +\infty[\varepsilon$ and $|\infty| \, C, F, \neg F$ are never satisfied.

If $[i, +\infty[F_1, \ldots, F_m, \neg F_{m+1}, \ldots, \neg F_n$ and $|\infty| \, C, \neg F_1, \ldots, \neg F_m, F_{m+1}, \ldots, F_n$ both hold, then there is an infinite sequence $k_0 < k_1 < \ldots$ of states where C and $\neg F_1 \wedge \ldots \wedge \neg F_m \wedge F_{m+1} \wedge \ldots \wedge F_n$ both hold, i.e. where C and $\neg F$ both hold, where F is $F_1 \vee \ldots \vee F_m \vee \neg F_{m+1} \vee \ldots \vee \neg F_n$. Take some k_p such that $k_p \geq i$: then $\neg F$ holds, but also F since $[i, +\infty[F$ holds. This justifies the rule in the bottom line of the figure.
Figure 5:
Consider the topmost resolution rule. If $[s, t]C, F$ and $[s', t']C', \neg F$ both hold under S, τ, then for each state k in the intersection of the intervals $[s, t]$ and $[s', t']$ (interpreted by τ), we both have C or F, and C' or $\neg F$. Whether F is true or false at k, C, C' holds at k, hence everywhere on the intersection of the intervals. To compute this intersection, we have four cases to consider, according to whether $s < s'$ or $s \geq s'$, and whether $t < t'$ or $t \geq t'$ under τ. This yields the four possible conclusions of the rule.

The argument is similar for the next three rules.

□

$$\frac{[s,t]\varepsilon}{s \geq t+1} \qquad \frac{[s,+\infty[\varepsilon}{\times} \qquad \frac{|\infty|\, C, F, \neg F}{\times}$$

$$\frac{[s,+\infty[F_1,\ldots,F_m,\neg F_{m+1},\ldots,\neg F_n \quad |\infty|\, C, \neg F_1,\ldots,\neg F_m, F_{m+1},\ldots,F_n}{\times}$$
$$(1 \leq n,\ 0 \leq m \leq n)$$

Here $\times$ is used as a marker for a closed branch.

Figure 4: Closing rules

$$\frac{\begin{array}{c}[s,t]C,F\\ [s',t']C',\neg F\end{array}}{\begin{array}{c|c|c|c} s \leq s'-1, t \leq t'-1 & s' \leq s, t \leq t'-1 & s \leq s'-1, t' \leq t & s' \leq s, t' \leq t \\ [s',t]C,C' & [s,t]C,C' & [s',t']C,C' & [s,t']C,C' \end{array}}$$

$$\frac{\begin{array}{c}[s,t]C,F\\ [s',+\infty[C',\neg F\end{array}}{\begin{array}{c|c} s \leq s'-1 & s' \leq s \\ [s',t]C,C' & [s,t]C,C' \end{array}} \qquad \frac{\begin{array}{c}[s,t]C,\neg F\\ [s',+\infty[C',F\end{array}}{\begin{array}{c|c} s \leq s'-1 & s' \leq s \\ [s',t]C,C' & [s,t]C,C' \end{array}} \qquad \frac{\begin{array}{c}[s,+\infty[C,F\\ [s',+\infty[C',\neg F\end{array}}{\begin{array}{c|c} s \leq s'-1 & s' \leq s \\ [s',+\infty[C,C' & [s,+\infty[C,C' \end{array}}$$

Figure 5: Resolution rules

We may represent constraint sets K in any way that we wish. A practical representation of K is as a directed graph, whose vertices are the time constants occurring in K, and whose edges from c to c' correspond exactly to constraints $(c \leq c' + n)$ in K, and are labeled by n. We may actually impose that there is at most one edge from c to c' by merging $c \xrightarrow{n} c'$ and $c \xrightarrow{n'} c'$ into $c \xrightarrow{n''} c'$, with $n'' = \min(n, n')$. Call a *path* in this graph any finite sequence $c_0 \xrightarrow{n_0} c_1 \xrightarrow{n_1} \ldots \xrightarrow{n_{k-1}} c_k$, $k \geq 0$; the *weight* of this path is $n_0 + n_1 + \ldots + n_{k-1}$; it is a *cycle* if and only if $k > 0$ and $c_k = c_0$. We have:

Lemma 5 *Let K be a well-formed constraint set, where "well-formed" means that, for every vertex c, there is a path from c_0 to c of non-positive weight.*

K is satisfiable if and only if it has no cycle of strictly negative weight.

Proof: This is mostly well-known [Bel58, Pra77]. A simple argument is as follows. If K has a cycle of negative weight, then any interpretation τ must satisfy $\tau(c_0) \leq \tau(c_k) + n_0 + \ldots + n_{k-1} < \tau(c_k) = \tau(c_0)$, which is impossible.

Conversely, if K has no cycle of negative weight, then for every two vertices c and c', either there is no path from c to c', or there is a path of least weight from c to c'. Define the *distance* $d(c, c')$ as $+\infty$ if there is no path from c to c', and as the weight of any path of least weight from c to c'. Let τ be the function mapping every vertex c to $-d(c_0, c)$. Since K is well-formed, there is a path from c_0 to c of non-positive weight w; by definition $d(c_0, c) \leq w \leq 0$, so $\tau(c)$ is a non-negative integer. Moreover, $\tau(c_0) = -d(c_0, c_0) = 0$ because there is a path of weight 0 from c_0 to c_0 (the trivial path), but there is no non-trivial path of negative weight from c_0 to c_0 (which would be a cycle); so τ is a time assignment.

Finally, for any edge $c \xrightarrow{n} c'$, the path of smallest weight from c_0 to c' ending in this edge has a weight that is no less than the path of smallest weight from c_0 to c', so $d(c_0, c) + n \geq d(c_0, c')$. Therefore $\tau(c) \leq \tau(c') + n$, so τ satisfies K. $\square$

Checking whether K is satisfiable therefore amounts to checking whether there is a cycle with negative weight in K: this is checkable in polynomial time by Floyd's or Ford's shortest path algorithm (see [McH90], Chapter 3); there are even efficient *incremental* algorithms to do this [AIMSN90]. It is easy to check that any constraint set produced by the tableau rules from the initial tableau $\mathcal{T}_0$ is well-formed.

So, as a last rule we introduce the *constraint rule* which declares a branch (B, K) "closed" (put the marker $\times$ on it) if K is unsatisfiable.

Theorem 6 *The resolution, closure and constraint rules are complete, i.e. given any unsatisfiable atomic branch (B, K), there is a finite tableau starting from (B, K) and produced using these rules, whose branches are all closed.*

Proof: The proof is by contraposition. Starting from (B, K), apply all resolution and closing rules until a saturated tableau $\mathcal{T}$ is reached, i.e. on each branch every rule that is applicable has been applied. $\mathcal{T}$ can be reached in finitely many steps. Assume that there is an open branch (B', K') in $\mathcal{T}$. We then claim that (B', K') has a model. Since $B \subseteq B'$ and $K \subseteq K'$, this is a contradiction.

First, because the set of constraints K' is satisfiable, let τ be a mapping from time constants to integers satisfying it, and let N be the highest integer in the range of τ. Let also $|\infty| C_1, \ldots, |\infty| C_n$ be the signed clauses with sign $|\infty|$ in B'. We build a structure (S, ξ) on which B' will hold.

For every i such that $0 \leq i \leq N$, let:

$$S_i = \{[s,t]C \in B' \mid \tau(s) \leq i \leq \tau(t)\} \cup \{[s, +\infty[C \in B' \mid \tau(s) \leq i\}$$

and let S'_i be the clauses in S_i without signs:

$$S'_i = \{C \mid I\ C \in S_i \text{ for some interval } I\}$$

S_i is closed under the resolution rules of Figure 5, hence S'_i is closed under the usual propositional resolution rule. By the completeness of propositional resolution:

- either S'_i contains the empty clause, and S_i then contains some clause $[s,t]\varepsilon$ or $[s, +\infty[\varepsilon$; since B' is open, the second case is impossible; in the first case, and since B' is saturated, K' must contain the constraint $s \geq t+1$, which contradicts our assumption that $j \leq i \leq k$, where j and k are the values of s and t under τ;

- or S'_i does not contain the empty clause, so S'_i then has a model: let ξ map i to the set of propositional variables that are true under this model.

For every $j \in \mathbb{N}$, for every $1 \leq i \leq n$, we build $\xi(N + jn + i)$ as follows. Let T_i be the set of all clauses of the form $[s, +\infty[C$ in B', and T'_i be the set of the corresponding C's. Again, T_i is closed under the resolution rules of Figure 5, hence T'_i is closed under the usual propositional resolution rule, and cannot contain the empty clause. Let C_i be written as $L_{i1}, \ldots, L_{in_i}$. Let then T''_i be T'_i union the n_i unit clauses $L_{i1}, \ldots, L_{in_i}$. Let T'''_i be the closure of T''_i under the propositional resolution rule. Since T'_i is already closed under this rule and by the closure rules from Figure 4, resolution among the units $L_{i1}, \ldots, L_{in_i}$ does not occur, and the only possible resolution steps take as one parent clause $C, \neg L_{ik_1}, \ldots, \neg L_{ik_j}$ in T'_i and resolve it, maybe repeatedly, with complementary units L_i from $L_{i1}, \ldots, L_{in_i}$. The empty clause could enter T''' only if a clause in T'_i consisted exclusively of complements of literals in $L_{i1}, \ldots, L_{in_i}$. But in this case the bottommost closing rule of Figure 4 would close branch B', which contradicts our assumption. T'''_i is therefore a resolution-closed set of clauses without the empty clause. Again by the completeness of propositional resolution, T'''_i then has a model: let ξ map $N + jn + i$ to the set of propositional variables that are true under this model.

Consider any clause of the form $[s,t]C$ in B'. We have $\mathbb{N}, \xi \models [s,t]C$. Indeed, for every i such that $s \leq i \leq t$ (modulo τ), in particular $0 \leq i \leq N$, and $[s,t]C$ is in S_i, so $C \in S'_i$, so by construction $\xi(i)$ makes C true.

Consider any clause $|\infty| C_i$, $1 \leq i \leq n$. By construction, every literal of C_i is true under ξ at state $N + jn + i$ (they are the clauses that we have added to T'_i to yield T''_i). So $\mathbb{N}, \xi, N + jn + i \models C_i$, hence $\mathbb{N}, \xi \models |\infty| C_i$.

Finally, consider any clause $[s, +\infty[C$ in B'. For every i such that $s \leq i \leq N$, C holds at state i: the argument is as for clauses of the form $[s,t]C$. For every state $N + jn + i$, $j \geq 0$, $1 \leq i \leq n$, by construction $\xi(N + jn + i)$ is a propositional model of T'''_i, hence of T'_i. Therefore C holds at state $N + jn + i$. To sum up, C holds at every state $\geq s$, hence $\mathbb{N}, \xi \models [s, +\infty[C$. $\square$

5.2 Refinements of Resolution

Because the proof of Theorem 6 rests on the completeness of propositional resolution in a rather simple way, it is tempting to use standard refinements of resolution [CL73].

For instance, consider *ordered resolution*: given a total ordering $>$ on literals, ordered resolution is defined by the rules of Figure 5, where in each rule, F must be the $>$-maximal literal in C and C'. (This is well-defined provided that we eliminate tautologies, i.e. clauses containing both G and $\neg G$ for some G.) But this does not preserve completeness; the branch:

$$\begin{array}{l}[c_0, +\infty[A, B \\ [c_0, +\infty[\neg A \\ |\infty| \neg B\end{array}$$

with $B > A$ is unsatisfiable, but no rule applies. The reason why the proof of completeness breaks in this case is that, although T_i' (we use throughout this discussion the notation from the previous proof) is closed under the (here, ordered) resolution rule, T_i''' is now not so easily obtained from T_i'' as before: in the example, T_i''' is the set of clauses (A, B), $(\neg A)$ and $(\neg B)$, from which we can deduce (A), then ε by ordered resolution. Similar arguments entail that hyper-resolution or linear resolution are also incomplete.

This is easily repaired by introducing new resolution rules that guarantee that any clause C in T''' occurs as $\sigma\ C$ on the saturated branch B' with some sign σ. A new sign has to be introduced for this purpose: for each signed clauses of the form $|\infty|\, C_i$, where $C_i = L_{i1}, \ldots, L_{in_i}$, we create a new sign $|\infty|_{C_i}$, and transform C_i into the n_i unit signed clauses:

$$\begin{array}{l}|\infty|_{C_i}\ L_{i1} \\ \ldots \\ |\infty|_{C_i}\ L_{in_i}\end{array}$$

We then close branches by using the resolution rules of Figure 5, the additional resolution rules of Figure 6 and the closing rules of Figure 7, which replace those of Figure 4. Semantically, $|\infty|_{C_i}\ C$, where C is a set of literals, means that the disjunction (not the conjunction!) of literals in C holds at all infinitely many time points $N + jn + i$, $j \geq 0$, of the proof of Theorem 6; at all these time points, the conjunction of literals in C_i holds.

$$\frac{[s, +\infty[C, F \quad |\infty|_{C_i}\ C', \neg F}{|\infty|_{C_i}\ C, C'} \qquad \frac{[s, +\infty[C, \neg F \quad |\infty|_{C_i}\ C', F}{|\infty|_{C_i}\ C, C'} \qquad \frac{|\infty|_{C_i}\ C, \neg F \quad |\infty|_{C_i}\ C', F}{|\infty|_{C_i}\ C, C'}$$

Figure 6: Additional resolution rules

$$\frac{[s, t]\varepsilon}{s \geq t + 1} \qquad \frac{[s, +\infty[\varepsilon}{\times} \qquad \frac{|\infty|_{C_i}\ \varepsilon}{\times}$$

Figure 7: Closing rules (bis)

Theorem 7 *For any complete refinement of resolution (ordered, hyper-resolution, linear resolution, etc.), the rules of Figures 5, 6 and 7 used according to the given refinement are complete.*

Proof: As for Theorem 6, replacing the argument at time points $N + jn + i$, $j \geq 0$, by the above discussion. □

The following standard tricks carry over from propositional resolution to our calculus. Let σ denote any sign, i.e. any interval I or any infinity sign $|\infty|_{C_i}$.

- Unit resolution: if there is a clause $\sigma\ L$, where σ is a sign and L is a literal, then resolve with this clause prioritarily;
- Pure clause elimination: eliminate clauses of the form $\sigma\ C, A$, such that $\neg A$ does not appear on the (atomic) branch; similarly, eliminate clauses of the form $\sigma\ C, \neg A$ if A does not appear on the branch;
- Subsumption: if there are two clauses $\sigma\ C$ and $\sigma'\ C'$ on the branch, with $C \subseteq C'$ and σ provably contains σ', then erase $\sigma'\ C'$; we say that $[s,t]$ provably contains $[s',t']$ if and only if $K \models t' \leq s'-1$, or $K \models s \leq s'$ and $K \models t' \leq t$, where K is the constraint set of the branch; ($K \models s \leq s'$ means by definition that $K \cup \{s \geq s'+1\}$ is satisfiable;) similarly, $[s,+\infty[$ provably contains $[s',t']$ if and only if $K \models t' \leq s'-1$, or $K \models s \leq s'$; $[s,+\infty[$ provably contains $[s',+\infty[$ if and only if $K \models s \leq s'$; $|\infty|_{C_i}$ provably contains $|\infty|_{C_j}$ if $C_i = C_j$; and these are the only cases.
- We may also, before we rewrite any formula $|\infty|C$ as a set of unit clauses, erase all such clauses that are not maximal, i.e. such that there is a larger signed clause $|\infty|C, C'$.

5.3 Complexity Issues

The proof of Theorem 2 shows that branches may become exponentially long. The problem lies in the rules with premise $[s,t]C, F \wedge G$ or similar ($[s,+\infty[C, F \wedge G$, $[s,t]C, \neg(F \vee G)$, $[s,+\infty[C, \neg(F \vee G)$), which duplicate C. We may replace such rules by introducing new propositional variables, like:

$$\frac{[s,t]C, F \wedge G}{\begin{array}{c}[s,t]C, \nu_{F\wedge G}\\ [c_0,+\infty[\neg\nu_{F\wedge G}, F\\ [c_0,+\infty[\neg\nu_{F\wedge G}, G\end{array}}$$

saying that $\nu_{F\wedge G}$, the new variable associated with the sub-formula $F \wedge G$, is equivalent over the full domain $[c_0,+\infty[$ to $F \wedge G$ (we have not added $[c_0,+\infty[\neg F, \neg G, \nu_{F\wedge G}$, which is in fact not needed; this is a well-known trick in the field of definitional clausal forms). Then the branches would have polynomial length. We leave it to the reader to check the details. In practice, we have not needed such a trick.

From the complexity viewpoint, resolution is one of the worst propositional satisfiability procedures. In fact, the branches generated by the resolution rules can still have exponential length, because there are exponentially many clauses built on a given number of literals. There may also be up to exponentially many such branches, which enumerate all possible refinements of the constraints in the set of constraints K.

This can be remedied as follows. Assume that we have an atomic branch (B,K) with K satisfiable, and we want to know whether (B,K) is satisfiable. Since K is satisfiable, it defines a partial ordering on all interval bounds s, t appearing on the branch B. Guess (in polynomial time) a total ordering $\leq$ on these interval bounds that is compatible with the constraints in K. (To avoid some technicalities, first rewrite s to t if K forces s and t to be equal.) Then, split each interval $[s,t]$ (resp. $[s,+\infty[$) occurring in B as a union of minimal intervals $[s,s_1]$, $[s_1,s_2]$, …, $[s_k,t]$ (resp. $[s_k,+\infty[$), where $s \leq s_1 \leq s_2 \leq \ldots \leq s_k$ $(\leq t)$; minimal here means that for no minimal interval $[s,t]$ (resp. $[s,+\infty[$) there is another interval bound u such that $s \leq u \leq t$ (resp. $s \leq u$). Reformulate the signed clauses in B as as many formulas with minimal intervals as needed; i.e., rewrite $[s,t]C$ as the clauses $[s,s_1]C$, $[s_1,s_2]C$, …, $[s_k,t]C$, and similarly for $[s,+\infty[C$. Then, B is satisfiable under the ordering $\leq$ if and only if:

- for each minimal interval $[s,t]$, the set of clauses with $[s,t]$ as sign is propositionally satisfiable;
- and for the (unique) minimal interval of the form $[s,+\infty[$, if we let S be the set of clauses with $[s,+\infty[$ as sign, then S is propositionally satisfiable, and has satisfying assignments that also satisfy C, for each signed clause of the form $|\infty|C$ on the branch B.

Since propositional satisfiability can be checked efficiently (say, by the Davis-Putnam procedure), this procedure seems attractive. However, it is based on guessing a total ordering $\leq$ on the interval bounds in B, and may produce many clauses with minimal intervals as signs (although still a polynomial number). It does not seem, therefore, to be really practical. However, combined with the trick of introducing fresh variables $\nu_{F\wedge G}$ to deal with conjunctions on each branch, this re-proves the classic result:

Theorem 8 *Satisfiability of linear temporal logic formulas built on* $\Box$ *and* $\Diamond$ *as only temporal operators is NP-complete.*

Proof: It is NP-hard, because it contains propositional satisfiability. On the other hand, it is in NP. Indeed, develop a tableau for the input formula, using the $\nu_{F\wedge G}$ trick. Instead of developing the whole tableau, guess an atomic branch in the fully developed tableau (it has polynomial size), then check its satisfiability by guessing a linear ordering on interval bounds as above, and then guessing a truth assignment for propositional variables on each minimal interval. The total size of the guessed objects is clearly polynomial, while checking whether every guessed truth assignment satisfies every concerned clause, and checking whether the guessed linear ordering satisfies the set of constraints on the guessed branch, can be done in polynomial time. $\Box$

6 Further Improvements

In this section, we present some further improvements of our calculus that are easily seen to preserve correctness and completeness, but prove to be very effective in an actual implementation, see Section 8.
Let us first mention the simplification rules in Figure 8. The conclusion part of all these rules is empty. So their application amounts to the deletion of their premise, which, as can be easily observed, will not contribute to the closure of a branch. They may be used either with the calculus of Subsection 5.1, or with the refinements of resolution of Subsection 5.2.

$$\frac{[s,t]C, F, \neg F}{} \qquad \frac{[s,+\infty[C, F, \neg F}{} \qquad \frac{]\infty]\,\varepsilon}{}$$

Figure 8: Simplification rules

Another shortcut in the application of the expansion rules of Figures 1, 2, and 3 is the case when the premise consists of just one formula, i.e. $C = \varepsilon$, which allows us to omit any parts of the conclusion referring to C. Thus:

$$\frac{[s,+\infty[\varepsilon, \Box F}{[s,+\infty[\varepsilon \;\Bigg|\; \begin{array}{c}[s,u-1]\varepsilon\\ [u,+\infty[F\\ s\leq u\\ u \text{ fresh}\end{array}} \qquad \text{is replaced by} \qquad \frac{[s,+\infty[\Box F}{\begin{array}{c}[u,+\infty[F\\ s\leq u\\ u \text{ fresh}\end{array}}$$

Consider now the following problem. Assume you wish to prove some formula of the form $F \wedge G$. Then, it is enough to prove F and G separately, and this is what a normal tableau system for classical logic would do, for instance. However, here this would be transformed into $[c_0]\neg F \vee \neg G$, which we then transform into $[c_0]\neg F, \neg G$. But in this case (when the current interval contains only one point), it is usually better to transform it into two conclusions $[c_0]\neg F$ and $[c_0]\neg G$. We may therefore add the following *case decomposition* rule:

$$\frac{[s]F_1, \ldots, F_n}{[s]F_1 \mid \ldots \mid [s]F_n}$$

Using this rule eagerly amounts to use it right after the rules with premise $[s,t]C, F \vee G$ or $[s,t]C, \neg(F \wedge G)$ when $s = t$.

Specializing the rules of Figure 1 to use the case decomposition rule eagerly yields new rules. Then, notice that if we start from $[c_0]F$, where F is the formula to prove, then all signed clauses of the form $[s]C$ that we shall ever need are such that C contains exactly one formula. This yields the rules of Figure 9. Notice also that we do not need to include the case decomposition rule explicitly any longer.

We can also improve the procedure in several other ways. First, we can simplify the calculus by maintaining formulas in negation normal form (i.e., with negations pushed inwards till they vanish or they apply to propositional variables). This is interesting because it already includes basic principles of duality through negation; for example, this shortens the proof of $\neg\Diamond\Box F \vee \Box\Diamond F$ significantly.

$$\frac{[s]F \wedge G}{\begin{array}{c}[s]F\\ [s]G\end{array}} \qquad \frac{[s]F \vee G}{[s]F \mid [s]G} \qquad \frac{[s]\neg(F \wedge G)}{[s]\neg F \mid [s]\neg G} \qquad \frac{[s]\neg(F \vee G)}{\begin{array}{c}[s]\neg F\\ [s]\neg G\end{array}} \qquad \frac{[s]\neg\neg F}{[s]F}$$

$$\frac{[s]\Box F}{[s,+\infty[F} \qquad \frac{[s]\Diamond F}{\begin{array}{c}[u]F\\ s \leq u\\ u \text{ fresh}\end{array}} \qquad \frac{[s]\neg\Box F}{\begin{array}{c}[u]\neg F\\ s \leq u\\ u \text{ fresh}\end{array}} \qquad \frac{[s]\neg\Diamond F}{[s,+\infty[\neg F}$$

Figure 9: Logical rules, single time points

Another plausible simplification is to erase each signed clause of the form $[s,t]C$ on the current branch such that K proves $s \geq t+1$ (i.e., such that K plus $s \leq t$ is unsatisfiable). The current rules only do this when C is the empty clause. This is not needed in the tableau rules, because such removals will in fact never happen. However, this may significantly help the resolution rules. Each resolution rule indeed creates new branches on which new constraints are added, which may in turn provide opportunities to erase some clauses as above. This is a form of (temporal) tautology elimination.

7 An Example

$$1\ [s_0,s_0]\neg(\neg\Box(\neg\Box(\neg P \vee \Box P) \vee P) \vee \neg\Diamond\Box P \vee P)$$

$$\begin{array}{ll}
2[1] & [s_0,s_0]\Box(\neg\Box(\neg P \vee \Box P) \vee P)\\
3[1] & [s_0,s_0]\Diamond\Box P\\
4[1] & [s_0,s_0]\neg P\\
5[2] & [s_1,+\infty[\neg\Box(\neg P \vee \Box P), P\\
6[2] & s_0 \leq s_1 \leq s_0\\
7[3] & [s_2,s_2]\Box P\\
8[3] & s_0 \leq s_2\\
9[7] & [s_3,+\infty[P\\
10[7] & s_2 \leq s_3 \leq s_2
\end{array}$$

$$\begin{array}{ll}
11[5] & [s_1,+\infty[P\\
12[11,4] & [s_0,s_0]\varepsilon\\
13[12] & s_0 \geq s_0+1\\
\times &
\end{array}$$

$$\begin{array}{ll}
14[5] & [s_4,s_4]\neg(\neg P \vee \Box P)\\
15[5] & [s_4,+\infty[P\\
16[5] & s_1 \leq s_4\\
17[14] & [s_4,s_4]P\\
18[14] & [s_4,s_4]\neg\Box P\\
19[18] & [s_5,s_5]\neg P\\
20[18] & s_4 \leq s_5\\
21[19,15] & [s_5,s_5]\varepsilon\\
22[21] & s_5 \geq s_5+1\\
\times &
\end{array}$$

$$\begin{array}{ll}
23[5] & |\infty| P, \neg\Box P\\
24[23] & |\infty| P\\
25[23] & |\infty|\neg P\\
26[25,9] & \times
\end{array}$$

Figure 10: Example, Dummett's formula

To illustrate how proof search proceeds, look at Figure 10: this is a proof of Dummett's formula $\Box(\Box(p \to \Box p) \to p) \to (\Diamond(\Box p) \to p)$. We use the shortcuts mentioned in Section 6. To help the reader each line in the proof is numbered, and the line numbers of the parent formulas are shown between brackets.

On the leftmost branch, we resolve formulas 11 and 4; since the constraints on the branch imply that $s_0 = s_1$, there is only one possible resolvent, namely $[s_0, s_0]\varepsilon$, which is unsatisfiable. On the middle branch, resolution of 19 and 15 yields the only resolvent $[s_5, s_5]\varepsilon$ since $s_4 \leq s_5$, and this resolvent is unsatisfiable. The rest can readily be read from Figure 10.

8 Experimental Results

We have implemented this system of logic in the HimML byte-coded interpreter, an implementation of Standard ML with fast set and map operations. The rules of Figure 9 were used, and we have chosen to always maintain formulas in negation normal form.

Our expansion strategy is one of *regular tableaux*: we have a current goal (a signed formula on the branch), which we expand by some tableau rule, giving new subgoals to expand; this is so unless the goal has become atomic, in which case we focus on another non-atomic signed formula on the current branch as new goal; when the branch becomes atomic, we call the resolution rules. We also accumulate constraints along the way, using the graph structure of Lemma 5 and declare the branch closed whenever the current set of constraints becomes unsatisfiable. Finally, the formula that we select in the current goal clause is the first non-modal formula that we find in it, or the first modal formula ($\Box F$ or $\Diamond F$) if there are no non-modal formulas left in the goal.

The resolution rules are only applied on atomic branches, i.e. on branches that contain signed clauses of the form $[s, t]C$, $[s, +\infty[C$ or $|\infty| C$, where C is a multiset of literals. We have used positive hyper-resolution, where (non-unit) clauses to resolve are chosen so that one of them is a positive clause, i.e. does not contain any negated atom; if there are unit clauses, we resolve prioritarily on them, whether they are positive or negative. This also means that we have used the closing rules of Figure 7 instead of Figure 4. No subsumption test has been used, except that no duplicate clause is ever added to the clause set. No other trick has been used.

We tested 46 formulas, 22 of which are valid in linear temporal logic: see Figure 11, where the valid formulas are checked with a $\surd$ sign. Non measurable times were written as – (under 0.017 s.) Measurement errors are around 0.017 s., if we overlook garbage collections, which can interrupt any computation at any time for a few tens of milliseconds. Tests were conducted under the HimML toplevel running on a diskless Sun 4 workstation under SunOS 4.1.2, with 16Mb core memory. The maximum process size was 2096 Kb, and garbage collection took 9.3% of the time. Total running time was 1.47 s.

The timings are good, and no combinatorial explosion occurs. But, for the most part, the benchmark formulas are not that big: none has more than two propositional variables, in particular. Of course, the relevant measure of complexity of a formula is not its number of variables, but rather the number of variables and of modal sub-formulas that it contains. In this respect, formulas like Dum or Grz already achieve a respectable size. But we are still far from real-life specifications, even from realistic toy problems [MP91]. All the realistic problems that we know involve a richer set of connectives, namely that of full LTL: they will be considered in a forthcoming paper. For now, all that we can say is that the ideas presented here seem to be a good start.

9 Conclusion

We have presented a tableau calculus for the fragment LTL_0 of linear temporal logic and shown its viability in a first implementation. There are two roads open from here.

- To extend the present approach to full linear temporal logic; this is under way and looks promising.
- To look for further improvements. Some improvements on the tableau extension part have already been proposed here. There are more techniques along these lines known from first-order logic (using lemmas, or structures like links to accelerate the detection of closed branches, and so on). It is certainly worthwhile to investigate if they can also be used with advantage in the context of linear temporal logic. It would be even more interesting to improve the second part of the algorithm, which tries to close branches. Two concrete questions may guide future research: Is it possible to adapt Floyd's or Ford's graph algorithm, which just detects negative cycles, to directly check the satisfiability of an atomic

Name	Statement	DAG size	size	time (s.)	
T	$\Box p \to p$	4	5	–	$\surd$
pb2	$p \to \Box p$	4	5	0.017	
G	$\Diamond(\Box p) \to \Box(\Diamond p)$	7	8	0.017	$\surd$
M	$\Box(\Diamond p) \to \Diamond(\Box p)$	7	8	0.033	
M2	$\Diamond(\Box(p \to \Box p))$	6	7	0.017	
M3	$\Box(\Diamond p) \wedge \Box(\Diamond q) \to \Diamond(p \wedge q)$	11	13	0.033	
pb5	$p \to \Diamond p$	4	5	0.017	$\surd$
lin1	$(\Diamond p \wedge \Diamond q) \to (\Diamond(p \wedge \Diamond q) \vee \Diamond(q \wedge \Diamond p))$	12	18	0.067	$\surd$
Dum	$\Box(\Box(p \to \Box p) \to p) \to (\Diamond(\Box p) \to p)$	13	18	0.050	$\surd$
Dum1	$\Box(\Box(p \to \Box p) \to p) \to (\Diamond(\Box p) \to \Box p)$	13	19	0.050	$\surd$
Dum2	$\Box(\Box(p \to \Box p) \to \Box p) \to (\Diamond(\Box p) \to p)$	13	19	0.083	$\surd$
Dum3	$\Box(\Box(p \to \Box p) \to \Box p) \to (\Diamond(\Box p) \to \Box p)$	13	20	0.083	$\surd$
Dum4	$\Box(\Box(p \to \Box p) \to p) \to (\Diamond(\Box p) \to p \vee \Box p)$	14	21	0.050	$\surd$
D	$\Box p \to \Diamond p$	5	6	–	$\surd$
4	$\Box p \to \Box(\Box p)$	5	7	0.017	$\surd$
4M	$\Box p \wedge \Diamond q \to \Diamond(\Box p \wedge q)$	9	12	0.017	$\surd$
5	$\Diamond p \to \Box(\Diamond p)$	5	7	0.017	
5M	$\Diamond p \wedge \Diamond q \to \Diamond(\Diamond p \wedge q)$	9	12	0.033	
B	$p \to \Box(\Diamond p)$	5	6	0.017	
BM	$p \wedge \Diamond q \to \Diamond(\Diamond p \wedge q)$	9	11	0.033	
G0	$\Diamond(p \wedge \Box q) \to \Box(p \vee \Diamond q)$	10	12	0.017	$\surd$
H	$\Box(p \vee q) \wedge \Box(\Box p \vee q) \wedge \Box(p \vee \Box q) \to \Box p \vee \Box q$	15	23	0.083	$\surd$
H+	$\Box(\Box p \vee q) \wedge \Box(p \vee \Box q) \to \Box p \vee \Box q$	12	18	0.083	$\surd$
L	$\Box(p \wedge \Box p \to q) \vee \Box(q \wedge \Box q \to p)$	13	17	0.033	$\surd$
L+	$\Box(\Box p \to q) \vee \Box(\Box q \to p)$	11	13	0.017	$\surd$
L++	$\Box(\Box p \to \Box q) \vee \Box(\Box q \to \Box p)$	11	15	0.033	$\surd$
Pt	$\Box(p \vee \Diamond p) \to \Diamond(p \wedge \Box p)$	9	12	0.033	
W	$\Box(\Box p \to p) \to \Box p$	7	10	0.017	
Z	$\Box(\Box p \to p) \to (\Diamond(\Box p) \to \Box p)$	10	15	0.033	
Grz	$\Box(\Box(p \to \Box p) \to p) \to p$	10	13	0.017	
Grz1	$\Box(\Box(p \to \Box p) \to p) \to \Box p$	10	14	0.050	
Grz2	$\Box(\Box(p \to \Box p) \to \Box p) \to p$	10	14	0.017	
Grz3	$\Box(\Box(p \to \Box p) \to \Box p) \to \Box p$	10	15	0.017	
Grz4	$\Box(\Box(p \to \Box p) \to p) \to p \vee \Box p$	11	16	0.050	
Grz5	$\Box(\Box(p \to \Box q) \to \Box q) \wedge \Box(\Box(\neg p \to \Box q) \to \Box q) \to \Box q$	18	28	0.033	
F	$(\Diamond(\Box p) \to q) \vee \Box(\Box q \to p)$	11	13	0.033	
Hs	$p \to \Box(\Diamond p \to p)$	7	9	0.017	
P	$\Diamond(\Box(\Diamond p)) \to p \to \Box p$	9	11	0.033	
R	$\Diamond(\Box p) \to p \to \Box p$	7	10	0.017	
X	$\Box(\Box p) \to \Box p$	5	7	0.017	$\surd$
Zem	$\Box(\Diamond(\Box p)) \to p \to \Box p$	8	11	0.017	
K1	$\Diamond p \to \Box p$	5	6	0.017	
K2	$\Box(p \vee q) \to \Box p \vee \Box q$	9	11	0.033	
K3	$\Diamond(\Box(\Diamond p)) \leftrightarrow \Box(\Diamond p)$	9	19	0.033	$\surd$
K4	$\Box(\Diamond(\Box p)) \leftrightarrow \Diamond(\Box p)$	9	19	0.017	$\surd$
K5	$\Box(\Diamond(p \vee q)) \leftrightarrow \Box(\Diamond p) \vee \Box(\Diamond q)$	15	29	0.050	$\surd$

Figure 11: Experimental Results

branch (B, K)? Is it possible to replace the resolution principle, which is known not to perform very well on propositional logic, by a variant of the Davis-Putnam-procedure, or a variant of BDD-based satisfiability algorithms?

References

[AIMSN90] G. Ausiello, G. F. Italiano, A. Marchetti Spaccamela, and U. Nanni. Incremental algorithms for minimal length paths. In *Proc. 1st ACM-SIAM Symposium on Discrete Algorithms*, pages 12–21, 1990.

[BBC+96] N. Bjørner, A. Browne, E. Chang, M. Colón, A. Kapur, Z. Manna, H. Sipma, and T. Uribe. STeP: Deductive-algorithmic verification of reactive and real-time systems. In *Proc. 8^{th} Conference on Computer Aided Verification*, 1996.

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[CL73] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics. Academic Press, 1973.

[Gei95] K. Geiss. Implementierung eines Beweisers für lineare Temporallogik in CLP-Technologie, April 1995.

[HI94] R. Hähnle and O. Ibens. Improving temporal logic tableaux using integer constraints. In *Proc. International Conference on Temporal Logic, Bonn, Germany*, volume 827 of *Lecture Notes in Computer Science*, pages 535 – 539. Springer Verlag, 1994.

[MAB+94] Z. Manna, A. Anuchitanukul, N. Bjørner, A. Browne, E. Chang, M. Colón, L. De Alfaro, H. Devarajan, H. Sipma, and T. Uribe. STeP: The Stanford Temporal Prover. Computer science report, Computer Science Department, Stanford University, 1994.

[McH90] J. A. McHugh. *Algorithmic Graph Theory*. Prentice-Hall International, 1990.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1991.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.

[Pra77] V. R. Pratt. Two easy theories whose combination is hard. Technical report, M.I.T., September 1977.

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

[Wol85] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28^{e} année, 110-111:119–136, 1985.