July 1989 Report No. STAN-CS-89-1267

A Really Temporal Logic

by

Rajeev Alur and Thomas A. Henzinger

Department of Computer Science

Stanford University

Stanford, California 94305

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE	Form Approved OMB No 0704-0188
1a REPORT SECURITY CLASSIFICATION	1 b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT
2b DECLASSIFICATION DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) STAN-CS-89-1267	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION DEPT. OF COMPUTER SCIENCE; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) STANFORD UNIV. STANFORD, CA 94305	7b ADDRESS (City, State, and ZIP Code) N00039-84-C-0211
8a NAME OF FUNDING / SPONSORING ORGANIZATION DARPA; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) ARLINGTON, VA 22209	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) A REALLY TEMPORAL LOGIC	
12 PERSONAL AUTHOR(S) RAJEEV ALUR, THOMAS A. HENZINGER	
13a TYPE OF REPORT; 13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day); 15 PAGE COUNT
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES: FIELD; GROUP; SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Abstract. We introduce a real-time temporal logic for the specification of reactive systems. The novel feature of our logic, TPTL. is the adoption of temporal operators as quantifiers over time variables: every modality binds a variable to the time(s) it refers to.

TPTL is demonstrated to be both a natural specification language as well as a suitable formalism for verification and synthesis. We present a tableau-based decision procedure and model-checking algorithm for TPTL. Several generalizations of TPTL are shown to be highly undecidable.

20 DISTRIBUTION AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION	
22a NAME OF RESPONSIBLE INDIVIDUAL ZOHAR MANNA	22b TELEPHONE (Include Area Code) (415) 723-2273	22c OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

A Really Temporal Logic[1,2]

Rajeev Alur Thomas A. Henzinger
Department of Computer Science
Stanford University

July 21, 1989

Abstract. We introduce a real-time temporal logic for the specification of reactive systems. The novel feature of our logic, TPTL, is the adoption of temporal operators as quantifiers over time variables; every modality binds a variable to the time(s) it refers to.

TPTL is demonstrated to be both a natural specification language as well as a suitable formalism for verification and synthesis. We present a tableau-based decision procedure and model-checking algorithm for TPTL. Several generalizations of TPTL are shown to be highly undecidable.

1 Introduction

Linear temporal logic has been demonstrated to be a suitable specification formalism for reactive systems and their behavior over time ([Pn77], [OL82], [MP89]). The tableau-based satisfiability algorithm for its propositional version, PTL, forms a proven basis for the verification and synthesis of finite-state systems ([LP84], [MW84]).

PTL is interpreted over models that abstract away from the actual times at which events occur, retaining only the temporal ordering information. The analysis of systems with hard real-time requirements, such as bounded response time, calls, however, for the development of formalisms with explicit time.

[1]This research was supported in part by an IBM graduate fellowship to the second author, by the National Science Foundation under grant CCR-8812595, by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211, and by the United States Air Force Office of Scientific Research under contract 88-0281.

[2]An abbreviated version of this paper appears in the proceedings of the 30th IEEE Symposium on *Foundations of Computer Science* (1989).

Several attempts have been made to introduce time explicitly in PTL, and interpret it over models that associate a time with every state ([BH81], [KVD83], [Os87], [Ha88], [NA88], [PH88]). Even though the logics developed there have been shown to be useful for the specification of real-time systems, most of the important decidability and complexity questions have not been answered. In particular, the class of timing constraints that may be permitted, in the finite-state case, without sacrificing (elementary) decidability has not been identified.

Our objective is to develop an elementary real-time extension of PTL that allows us to generalize the PTL-based verification and synthesis tools. To begin with, a notational extension of PTL is required to be able to relate the times associated with different states. One commonly proposed method is to employ first-order temporal logic, with one of the state variables representing time ([Os87], [Ha88]). We claim that the unconstrained quantification of global variables allowed by this approach does not restrict the user to reasonable and readable specifications.

We propose a novel, restricted, form of quantification — we call it *temporal quantification* -, in which every variable is bound by a temporal operator and ranges over the times of the states to which the operator refers. This makes the presence of a dynamic time variable superfluous. Temporal quantification identifies, so we argue, precisely the subclass of "natural," intended specifications, and it leads to a concise and readable notation. For instance, a typical bounded-response property (that every p-state is followed by a q-state within ten time units) can be simply stated as

$$\Box x.(p \rightarrow \Diamond y.(q \wedge y \leq x + 10)).$$

Secondly, we need to identify how expressive a theory of time may be added, in this fashion, to PTL without sacrificing its elementary complexity. Our main results are twofold; we develop a near-optimal decision procedure and model-checking algorithm for real-time PTL, by restricting both the syntax and semantics of the time component, and show that these restrictions cannot be relaxed without losing decidability.

In particular, adding the theory of the natural numbers with successor, $\leq$, and congruences to PTL, yields an EXPSPACE-complete real-time temporal logic — TPTL. The tableau method for PTL can be generalized to TPTL. However, allowing either addition over time, or dense models, thus still combining two decidable formalisms, results in highly undecidable (Π^1_1-complete) logics.

Thus we lay out a theoretical basis for the verification and synthesis of finite-state real-time systems and, simultaneously, identify a fundamental boundary between the decidability and undecidability of finite-state formalisms with explicit time.

2 Timed Temporal Logic

We define timed PTL, TPTL, and demonstrate its adequacy and naturalness as a real-time specification language.

2.1 Syntax of TPTL

We are given infinite sets P and V of propositions (p, q, . . .) and variables (x, y, . . .), respectively.

As atomic formulas we permit propositions and atoms of the (0, S, $\leq$, $\equiv_c$) fragment of arithmetic, where x $\equiv_c$ y denotes that x is congruent to y modulo the constant c. Atomic formulas are combined by propositional connectives and *temporal quantifiers* — that is, temporal operators that bind variables.

DEFINITION **[Syntax].** The terms π, atomic formulas α, and formulas ϕ of TPTL are inductively defined as follows:

$$\pi := \mathbf{x} \mid 0 \mid S^c\pi$$
$$\alpha$$
$$\phi := q \,\|\, \pi\,\text{fals} \leq \pi_2 \phi_1 | \pi_1 \equiv \phi \pi_2 | \bigcirc x.\phi \mid [x_1.\phi_1]\mathcal{U}[x_2.\phi_2]$$

for x $\in$ V, $p \in P$, and c = 1, 2, . . . •

We abbreviate $\bigcirc x.\ \phi$ to $\bigcirc\phi$ if x does not occur freely in ϕ; $\mathcal{U}$ is handled analogously. The abbreviations =, <, >, $\geq$, **true,** A, V , and $\leftrightarrow$ are defined as usual; informally we often write $x + c$ and c for $S^c x$ and $S^c 0$, respectively. Also, Ox. ϕ and $\Box x.\ \phi$ stand for $\mathbf{true}\mathcal{U}[x.\ \phi]$ and $\neg\Diamond x.\ \neg\phi$, respectively.

2.2 Semantics of TPTL

The **formulas of TPTL** are interpreted over timed state sequences — that is, **state** sequences whose states are associated with times from the discrete time domain TIME = **N.**

At this point, a few words about our model are in order. We do not use time simply as a state counter (and, thus, not as a notational variant for the next-operator $\bigcirc$ of PTL). The time between successive states is only required not to decrease; it may remain the same, or increase by an arbitrary amount . While strictly increasing time is sufficient to model synchronous systems. all of whose events occur at discrete clock ticks, the events of asynchronous systems take place over a dense time domain. We show, in a later section, that reasoning about truly dense time is, however, highly undecidable.

So we still adopt a discrete model of time, but we allow, instead, state changes to occur between clock ticks; successive states may be observed in a particular order, yet at the same (discrete) time. This can be viewetl, alternatively, as constraining TPTL to observe the real, dense time of events only with finite precision. Let us be more formal.

DEFINITION [Timed **state sequence].** A *state sequence* $\sigma = \sigma_0\sigma_1\sigma_2 \ldots$ is an infinite sequence of states $\sigma_i \subseteq P$, $i \geq 0$.

A time sequence $\tau = \tau_0\tau_1\tau_2 \ldots$ is an infinite sequence of times $\tau_i \in TIME$, $i \geq 0$, such that

(*i*) [*initiality*] $\tau_0 = 0$,
(*id*) [*monotonicity*] Vi $\geq$ 0. $\tau_i \leq \tau_{i+1}$, and
(iii) [progress] $\forall t \in$ TIME. $\exists i \geq 0$. $\tau_i > t$.

A timed state sequence ρ = (a, τ) is a pair consisting of a state sequence a and a time sequence τ. ■

The *initiality* condition ensures that the term 0 refers to the initial time, thus prohibiting specifications from referring to absolute time values.[3] The progress condition is also sensible for any real-time specification language. *Only monotonicity,* however, is required for the complexity results we obtain in later sections.

By σ^i $[\tau^i]$ we denote the suffix of the state [time] sequence σ $[\tau]$ that starts at state σ_i [time τ_i]. Furthermore, let ρ^i = (a', τ^i).

DEFINITION **[Satisfiability].** The timed state sequence $\rho = (\sigma, \tau)$ is a *model* of the TPTL-formula ϕ iff $\rho^0 \models_{\mathcal{E}_0} \phi$ for the initial environment $\mathcal{E}_0$: $V \rightarrow \{\tau_0\}$, where the truth predicate $\models$ is inductively defined as follows:

$\rho^i \models_{\mathcal{E}} p$ iff $p \in \sigma_i$
$\rho^i \models_{\mathcal{E}} \pi_1 \leq \pi_2$ iff $\mathcal{E}(\pi_1) \leq \mathcal{E}(\pi_2)$,
for $\mathcal{E}(0) := 0$ and $\mathcal{E}(S^c\pi) := \mathcal{E}(\pi) + c$

[3]The expressiveness of the constant 0 and condition (*i*), which allow reference to the initial time in closed formulas, could, alternatively, be obtained by a new temporal operator *Now*. In fact, *Now* would have to be the only temporal quantifier.

$\rho^i \models_{\mathcal{E}} \pi_1 \equiv_c \pi_2$ iff $\mathcal{E}(\pi_1) \equiv \mathcal{E}(\pi_2)$ modulo c
$\rho^i \not\models_{\mathcal{E}}$ **false**
$\rho^i \models_{\mathcal{E}} \phi_1 \rightarrow \phi_2$ iff $\rho^i \models_{\mathcal{E}} \phi_1$ implies $\rho^i \models_{\mathcal{E}} \phi_2$
$\rho^i \models_{\mathcal{E}} \bigcirc x.\ \phi$ iff $\rho^{i+1} \models_{\mathcal{E}[x \leftarrow \tau_{i+1}]} \phi$
$\rho^i \models_{\mathcal{E}} [x_1.\ \phi_1]\mathcal{U}[x_2.\ \phi_2]$ iff $\rho^j \models_{\mathcal{E}[x_2 \leftarrow \tau_j]} \phi_2$ for some j $\geq i$ and
$\rho^k \models_{\mathcal{E}[x_1 \leftarrow \tau_k]} \phi_1$ for all $i \leq k < j$.

(Here $\mathcal{E}[x \leftarrow t]$ denotes the environment that agrees with $\mathcal{E}$ on all variables except x, which is mapped to $t \in$ *TIME.)*

The formula ϕ is *satisfiable [valid]* iff some [every] timed state sequence is a model of ϕ. ■

Intuitively speaking, every temporal operator acts also as a quantifier and binds the associated variable to the time(s) it refers to. For instance, Ox. $\phi(x)$ holds in a model ρ iff $\phi(\tau_i)$ holds in some suffix ρ^i, $i \geq 0$, of ρ; that is, 0 binds x to the time of the state at which ϕ "eventually" holds. Similarly, Ox. $\phi(x)$ holds in ρ iff $\phi(\tau_i)$ holds in *every* suffix ρ^i, $i \geq 0$, of p.

2.3 TPTL as a specification language

We demonstrate how **TPTL** improves, by linking its time references to temporal operators, the readability of real-time specifications.

A typical hard real-time requirement for a reactive system is that a switch p has to be turned off (represented by the proposition q) within, say, ten time units of its activation. In **TPTL** this condition can be expressed by the formula

$$\Box x.(p \rightarrow p\mathcal{U}[y.(q \wedge Y \leq x + 10)]). \qquad (1)$$

Using conventional temporal operators and a dynamic (state) variable T that assumes the value of the current time in every state, this specification is usually written as follows ([Os87]):

$$\Box(p \wedge T = x \rightarrow p\mathcal{U}(q \text{ A } T = y \text{ A } y \leq x + 10)). \qquad (2)$$

Here x and y are auxiliary, global variables ranging over $TIME$.

The meaning of this formula depends, not surprisingly, on the quantification of x and y, which is left implicit. The very fact that the quantification is often omitted ([Os87], [PH88]) suggests that the authors have some particular quantifiers for x and y in mind, whose force, location, and order are considered to be so obvious that they are not worth mentioning. If any quantifiers are given, they form a prefix to the entire formula ([Ha88]).

We claim that the following quantification is the (only) "intended" one:

$$\Box\, \forall x.\,(p \land T = x \to p\,\mathcal{U}[\exists y.\,(q \land T = y \land y \le x + 10)]). \qquad (3)$$

Note that (3) is, if interpreted over timed state sequences, equivalent to (1), but not to (2) with any quantifier prefix; in particular it does not imply the stronger condition

$$\forall x.\, \exists y.\, \Box(p \land T = x \to p\,\mathcal{U}(q \land T = y \land y \le x + 10)). \qquad (4)$$

The difference is subtle: while (3) asserts that every p-state of time x is followed by p-states and. eventually, a q-state of time $y \le x+10$, (4) demands more; that if there is a p-state of time x, then there is a time $y \le x + 10$ such that every p-state of time x is followed by p-states and, eventually, a q-state of time y. For instance, the timed state sequence $\rho = (\sigma, \tau)$, where $\sigma = \{p\}, \{q\}, \{p\}, \{q\}, \{\}, \{\}, \ldots$ and $\tau = 0, 0, 0, 1, 2, 3, \ldots$, satisfies (3) but not (4).

Thus, TPTL may, alternatively, be viewed as a first-order temporal logic with a dynamic time variable T and global variables that are employed in a very restricted way: every global variable is bound to the ("current") value of T immediately upon introduction; it is associated with a temporal operator, and refers to the same set of states (i.e., their times). In particular, $\Box x.\,\phi, \Diamond x.\,\phi, [x_1.\,\phi_1]\,\mathcal{U}[x_2.\,\phi_2]$, and $\bigcirc x.\,\phi$ have exactly the meaning of

$$\Box \forall x.\,(T = x \to \phi),$$
$$\Diamond \exists x.\,(T = x \land \phi),$$
$$[\forall x_1.\,(T = x_1 \to \phi_1)]\mathcal{U}[\exists x_2.\,(T = x_2 \land \phi_2)],$$

and $\bigcirc \exists x.\,(T = x \land \phi)$, respectively. Note that the choice of universal versus existential quantifiers is a matter of taste, since $\forall x.\,(T = x \to \phi)$ is equivalent to $\exists x.\,(T = x \land \phi)$.

In fact, TPTL allows us to express timing constraints by concise and readable specifications (compare (1) with (3)) and yields to a tableau- based decision procedure in an extremely straightforward way (as we shall see) precisely because all of its global variables are associated with operators and, thus, "temporal" sets of states.

3 Timed Tableaux

We present a doubly-exponential- time, tableau-based decision procedure for TPTL, and show that the satisfiability problem for TPTL is EXPSPACE-complete. We then demonstrate how the tableau techniques can be applied to verify TPTL-properties of real-time systems.

3.1 Decision procedure for **TPTL**

Let us say that a timed state sequence ρ = *(a, τ)* is *Ii-bounded,* for a constant $K \in$ *TIME,* iff

$$\forall i \geq 0.\, \tau_i \leq \tau_{i+1} \leq \tau_i + K,$$

that is, the time increases from a state to its successor state by no more than K. To begin with, we restrict ourselves to K-bounded models for checking satisfiability.

This case has finite-state character: the times associated with states can be modeled by finitely many new propositions $CLOCK_t$, $0 \leq t \leq K$, which represent the time differences t between successive states. In particular, we capture the (state and) time information in ρ by the state sequence $\hat{\sigma}$ with

$$\hat{\sigma}_i = \sigma_i \cup \{\text{CLOCK,,+,-.,}\}$$

for all $i \geq 0$, which allows us to adopt the tableau techniques for PTL. Later, we show how we can find an appropriate constant K for any given T PTL-formula ϕ.

Following the usual presentation of the tableau-based decision procedure for PTL ([Wo83]), we show how to construct the initial tableau for ϕ. Checking the satisfiability of ϕ may then be reduced to checking whether the finite initial tableau for ϕ contains certain infinite paths. The tableau method for PTL is, in fact, included in our procedure as the special case in which ϕ contains no timing constraints.

We use. for the sake of simplicity, a single free variable. *T,* to denote the initial time; this can be easily achieved by renaming, in ϕ, all bound occurrences of *T,* and then replacing all occurrences of free variables and 0 by *T*.

We also assume that ϕ is built using the connectives A, $\vee$, and the temporal operators $\bigcirc$, 0, and 0 over atomic formulas and negated atomic formulas. All atoms are either propositions, or of one of the forms x $\leq S^c y$, $S^c x \leq$ y, and $x \equiv_{c'} S^c y$, for c' > c $\geq$ 0. Any TPTL-formula can clearly be transformed into this form by at most doubling its length. The extension of the method to accomodate the until-operator $\mathcal{U}$ should be obvious.

3.1.1 Updating timing constraints

The ·y observation underlying the tableau method for PTL is that any formula can be split into two conditions: a non- temporal requirement on the initial state and a temporal requirement on the rest of the model (i.e., the successor state). Since the number of conditions generated in this way is finite, checking for satisfiability is reducible to checking for satisfiability in a finite structure, the initial tableau.

The splitting of TPTL-formulas into a present and a future (nest-state) condition demands more care; to obtain the requirement on the successor state, all timing constraints need to be translated appropriately to account for the time increase t from the initial state to its successor. Consider, for example, the formula Ox. $\phi(x, T)$, and recall that the free occurrences of T are references to the initial time. This eventuality can be satisfied either by having the initial state satisfy $\phi(T, T)$, with all free occurrences of x replaced by T, or by having the next state satisfy the updated eventuality Ox. $\phi(x, T-t)$.

For $t > 0$, a naive replacement of T by T-t would, however, successively generate infinitely many new conditions. Fortunately, the *monotonicity* of time can be exploited to keep the tableau finite; the observation that x is always instantiated, in the "future," to a value greater than or equal to T, allows us to simplify timing assertions of the form $T \leq S^c x$ and $S^c x \leq T$ to **true** and **false,** respectively. We define, therefore, the formula ϕ^t that results from updating all time references T in ϕ as follows.

DEFINITION [Translation of time references]. Given a TPTL-formula ϕ and $t \in TIME$, the formula ϕ^t is defined inductively: $\phi^0 = \phi$; and ϕ^{t+1} is obtained from ϕ^t by replacing all terms of the form $S^c T$ $(c > 0)$ by $S^{c-1}T$, and all subformulas of the form $T \leq S^c x$, $S^c x \leq T$, and $T \equiv_{c'} S^c x$ $(c \geq 0)$ **by true, false,** and $T \equiv_{c'} S^{(c+1) \bmod c'} x$, respectively. •

The following lemma shows that this translation has the intended effect.

LEMMA [Change of initial time]. *Let $\rho = (\sigma, \tau)$ be a timed state sequence, $\mathcal{E} : V \to TIME$ an environment, $i \geq 0$, and $t \in TIME$ such that $t \leq \tau_i$ and $t \leq \mathcal{E}(x)$ for all* x $\in$ *V. For any* TPTL-formula ϕ, $\rho^i \models_{\mathcal{E}[T \leftarrow t]} \phi$ *iff* $\rho^i \models_{\mathcal{E}[T \leftarrow \tau_i]} \phi^{\tau_i - t}$. ∎

PROOF. The proof proceeds by a straightforward induction on the structure of ϕ. ∎

3.1.2 Closure of a TPTL-formula

The closure of a formula ϕ collects all conditions that may arise by recursively splitting ϕ into its present and future parts.

DEFINITION [Closure]. The *closure* $Cl(\phi)$ of a TPTL-formula ϕ is the smallest set containing ϕ that is closed under the operation *Sub,* which is defined as follows:

$$
\begin{aligned}
&Sub(\phi_1 \wedge \phi_2) := \{\phi_1, \phi_2\}\\
&Sub(\phi_1 \vee \phi_2) := \{\phi_1, \phi_2\}\\
&Sub(\bigcirc x.\, \phi(x)) := \{\phi^t(T) \mid 0 \leq t \leq K\}\\
&Sub(\Box x.\, \phi(x)) := \{\phi(T), \bigcirc\Box x.\, \phi(x)\}\\
&Sub(\Diamond x.\, \phi(x)) := \{\phi(T), \bigcirc\Diamond x.\, \phi(x)\}. \blacksquare
\end{aligned}
$$

The size of the closure of ϕ clearly depends on the constants contained in ϕ. We say that a constant c > 0 occurs in a TPTL-formula ϕ iff ϕ contains a subformula of the form x $\leq S^{c-1}y$ or $S^{c-1}x \leq$ y, or the predicate symbol $\equiv_c$.

LEMMA [Size of closure]. *Let $n - 1$ be the number of connectives (propositional and temporal) in ϕ, and k the product of all constants occurring in ϕ. Then $|Cl(\phi)| \leq 2nk$.* $\blacksquare$

PROOF. Given a formula ϕ, we define, by induction on the structure of ϕ, the set D_ϕ of formulas that contains ϕ and is closed under updating timing constraints: the set C_ϕ is, in addition, closed under subformulas:

$$
\begin{aligned}
&C_p = D_p := \{p\}\\
&C_{x \leq S^c y} = D_{x \leq S^c y} := \{x \leq S^c y, x \leq S^{c-1} y, \ldots x \leq y\}\\
&c_{x \equiv_{c'} S^c y} = D_{x \equiv_{c'} S^c y} := \{x \equiv_{c'} S^{c'-1} y, x \equiv_{c'} S^{c'-2} y, \ldots x \equiv_{c'} y\}\\
&c_{\neg\alpha} = D_{\neg\alpha} := \neg D_\alpha\\
&C_{\phi_1 \wedge \phi_2} := C_{\phi_1} \cup C_{\phi_2} \cup D_{\phi_1 \wedge \phi_2}, D_{\phi_1 \wedge \phi_2} := D_{\phi_1} \wedge D_{\phi_2}\\
&C_{\bigcirc x.\phi(x)} := C_{\phi(T)} \cup D_{\bigcirc x.\, \phi(x)}, D_{\bigcirc x.\, \phi(x)} := \bigcirc x.\, D_{\phi(x)}\\
&c_{\Box x.\phi(x)} := C_{\phi(T)} \cup D_{\Box x.\, \phi(x)} \cup D_{\bigcirc\Box x.\, \phi(x)}, D_{\Box x.\, \phi(x)} := \Box x.\, D_{\phi(x)}.
\end{aligned}
$$

(Here $\neg E = \{\neg\phi \mid \phi \in E\}$ for any set E of formulas; the other connectives are applied to sets in an analogous fashion.) The cases $S^c x \leq$ y, $\phi_1 \vee \phi_2$, and Oz. $\phi($ x) are treated similarly. Furthermore, let $E^\bullet = E \cup$ **{true, false}.**

Observe that $D_\phi \subseteq C_\phi$. It is straightforward to show, by induction on the structure of ϕ, that

(i) $\phi \in D_\phi$ and. hence, $\phi \in C_\phi$:

(ii) for all $t \geq 0$, $\phi^t(T) \in D^*_{\phi(T)}$ and, therefore, $\phi^t(T) \in C^*_{\phi(T)}$;

(iii) C^*_ϕ is closed under Sub (use (ii)).

From (i) and (iii) it follows that $Cl(\phi) \subseteq C^*_\phi$.

Thus. it suffices to show that $|D_\phi| \leq k$ and $|C_\phi| \leq 2nk$, which may again be done by induction on the structure of ϕ. •

3.1.3 Initial tableau of a TPTL-formula

Tableaus for **TPTL** are finite. directed state graphs (Kripke structures) with local and global consistency constraints on all states. The states are represented by sets of formulas that are closed under "subformulas," expressing conditions on the current state and the successor states. Every state contains, in addition, a proposition $CLOCK_t$, $0 \leq t \leq K$, which denotes the time difference to the successor states. Both requirements are incorporated in the following definition of local consistency.

DEFINITION [Consistency]. A set Φ of TPTL-formulas is *consistent* iff it satisfies the following conditions:

- $CLOCK_t$ is in Φ for precisely one $0 \leq t \leq K$; this $t \in$ TIME is referred to as $Clock(\Phi)$.
- **false** is not in Φ.
- At most one of p and $\neg p$ is in Φ.
- If ϕ_1 A ϕ_2 is in Φ, then so are ϕ_1 and ϕ_2.
- If ϕ_1 v ϕ_2 is in Φ, then so is at least one of ϕ_1 and ϕ_2.
- If $\Box x.\ \phi(x)$ is in Φ, then so are $\phi(T)$ and $\bigcirc\Box x.\ \phi(x)$.
- If Ox. $\phi(x)$ is in Φ, then so is at least one of $\phi(T)$ and $\bigcirc\Diamond x.\ \phi(x)$.
- If $T \sim S^cT$ is in Φ, then $0 \sim$ c holds in ☠ (for $\sim$ one of $\leq$, $\geq$, $\equiv_{c'}$, or its negation). ■

Now we are ready to define the initial tableau of a formula in a way that ensures the global consistency of both temporal and real-time constraints.

DEFINITION [Initial tableau]. Let $TCl(\phi) = Cl(\phi)$ U $\{CLOCK_t \mid 0 \leq t \leq$ K$\}$. The initial *tableau* $\mathcal{T}(\phi)$ for the TPTL-formula ϕ is a directed graph whose vertices are the (maximal) consistent subsets of $TCl(\phi)$, and which contains an edge from Φ to Ψ iff $\phi^{Clock(\Phi)}(T) \in \Psi$ for all $\bigcirc x.\ \phi(x) \in \Phi$. ■

The significance of the (finite) initial tableau $\mathcal{T}(\phi)$ for the formula ϕ is that every model of ϕ corresponds to an infinite path through $\mathcal{T}(\phi)$ along which all eventualities are satisfied, and vice versa. This implies a finite-model property for TPTL, in the sense that every satisfiable TPTL-formula

ϕ is satisfiable by a model whose state part. estended by the new propositions $CLOCK_t$, consists of only finitely many distinct states. Let us be more precise.

DEFINITION [ϕ-**path**]. An infinite path $\Phi = \Phi_0\Phi_1\Phi_2\ldots$ through a tableau satisfies the *progress* condition for time iff $Clock(\Phi_i) > 0$ for infinitely many $i \geq 0$. The path Φ is a *ϕ-path* iff, in addition, $\phi \in \Phi_0$ and for every $i \geq 0$, Ox. $\psi(x) \in \Phi_i$ implies $\psi^t(T) \in \Phi_j$ for some $j \geq i$ with $t = \Sigma_{i \leq k < j} Clock(\Phi_k)$. ■

We can characterize the length of +-paths by reducing every +-path to a special form. This will prove to be important to obtain an upper bound for the complexity of TPTL.

LEMMA **[Length of &paths].** *If a tableau with m vertices contains a &path, then it contains a ϕ-path of the form*

$$\Phi_0 \rightarrow \Phi_1 \rightarrow \ldots \rightarrow \Phi_k \rightarrow (\Phi_{k+1} \rightarrow \ldots \rightarrow \Phi_l)^\omega$$

for $1 \leq (2n+1)m$, where n is the number of connectives in ϕ. ■

PROOF. Consider the (infinite) \$-path $\Phi = \Phi_0\Phi_1\ldots$, and choose k to be the smallest i such that Φ_i occurs infinitely often in Φ. Now Φ_k contains at most n eventualities ψ_j, each one of which is satisfied by some vertex Ψ_j of $\Phi_k\Phi_{k+1}\ldots$

Let $\Phi^0 = \Phi_0 \ldots \Phi_k$, $\Phi^{2j-1} = \Phi_k \ldots \Psi_j$, and $\Phi^{2j} = \Psi_j \ldots \Phi_k$ $(1 \leq j \leq n)$ be finite segments of Φ that contain no other (i.e., inner) occurrences of Φ_k. Delete any loops in every Φ^i, thus obtaining $\hat{\Phi}^i$, $0 \leq i \leq 2n$, each of length at most m + 1. It is not hard to see that the result of deleting duplicated states (i.e., Φ_k) from

$$\hat{\Phi}^0(\hat{\Phi}^1\hat{\Phi}^2\ldots\hat{\Phi}^{2n})^\omega$$

is a ϕ-path of the desired form. •

3.1.4 Tableau decision procedure

The following main lemma suggests a decision procedure for TPTL: to determine if the TPTL-formula ϕ is satisfiable, construct the initial tableau $\mathcal{T}(\phi)$ and check whether it contains any @-paths.

LEMMA **[Correctness of initial tableau].**

(*a*) [soundness] *If $\mathcal{T}(\phi)$ contains a ϕ-path, then ϕ is satisfiable.*

(*b*) [completeness] *If ϕ is satisfiable (in a K-bounded model), then $\mathcal{T}(\phi)$ contains a ϕ-path.* •

The proof makes essential use of the *change-of-initial-time* lemma, and is more tedious than enlightening. Together with the *length-of-&paths* lemma it implies, in fact, that every satisfiable TPTL-formula is, in the sense mentioned above, satisfiable in a doubly exponential model.

PROOF. (n) Given a +-path $\Phi = \Phi_0\Phi_1 \ldots$ through $\mathcal{T}(\phi)$, define the timed state sequence $\rho = (\sigma, \tau)$ such that, for all i $\geq$ 0, $p \in \sigma_i$ iff $p \in \Phi_i$, and $\tau_{i+1} = \tau_i + Clock(\Phi_i)$. Note that τ satisfies the *progress* condition because Φ does. We show that, for all $i \geq 0$, $\psi \in \Phi_i$ implies $\rho^i \models_{[T \leftarrow \tau_i]} \psi$, by induction on the structure of ψ. Since $\phi \in \Phi_0$, it follows that ρ is a model of ϕ.

For a proposition $p \in \Phi_i$, we have $p \in \sigma_i$, and hence $\rho^i \models p$. If $\neg p \in \Phi_i$, then $p \notin \Phi_i$ because of the consistency of Φ_i; therefore $p \notin \sigma_i$ and $\rho^i \models \neg p$. If $T \sim S^cT \in \Phi_i$, then 0 $\sim$ c by the consistency of Φ_i, and therefore $\rho \models T \sim S^cT$. This completes the base cases.

If ψ_1 A $\psi_2 \in \Phi_i$, then also $\psi_1, \psi_2 \in \Phi_i$ because of the consistency of Φ_i. By the induction hypothesis, $\rho^i \models_{[T \leftarrow \tau_i]} \psi_1$ and $\rho^i \models_{[T \leftarrow \tau_i]} \psi_2$; hence $\rho^i \models_{[T \leftarrow \tau_i]} \psi_1$ A ψ_2. The disjunctive case is established similarly.

NOW assume that $\bigcirc x. \psi(x) \in \Phi_i$. Let $t = Clock(\Phi_i)$; then $\psi^t(T) \in \Phi_{i+1}$ and $\tau_{i+1} = \tau_i + t$. By the induction hypothesis, $\rho^{i+1} \models_{[T \leftarrow \tau_{i+1}]} \psi^t(T)$, that is, $\rho^{i+1} \models_{[T, x \leftarrow \tau_i + t]} \psi^t(x)$, and therefore $\rho^{i+1} \models_{[T \leftarrow \tau_i, x \leftarrow \tau_{i+1}]} \psi(x)$ *by* the *change-of-initial-time* lemma. Hence we can conclude that $\rho^i \models_{[T \leftarrow \tau_i]} \bigcirc x. \psi(x)$.

For the case that Ox. $\psi(x) \in \Phi_i$, we first show, by induction on j, that ox. $\psi^{\tau_j - \tau_i}$(x) $\in \Phi_j$ for all $j \geq i$. Note that $\tau_j - \tau_i = \Sigma_{i \leq k < j} Clock(\Phi_k)$ by our choice of τ. So suppose that $\Box x. \psi^{\Sigma_i \leq k < j} {}^{Clock(\Phi_k)}(x) \in \Phi_j$; then, by the consistency of Φ_j, also $\bigcirc\Box x. \psi^{\Sigma_i \leq k < j} {}^{Clock(\Phi_k)}(x) \in \Phi_j$ and, since $\mathcal{T}(\Phi)$ is a tableau, $\Box x. \psi^{\Sigma_i \leq k < j+1} {}^{Clock(\Phi_k)}(x) \in \Phi_{j+1}$.

We conclude therefore, again because of the consistency of Φ_j, that $\psi^{\tau_j - \tau_i}(T) \in \Phi_j$ for all j $\geq$ i. Applying the induction hypothesis, we obtain $\rho^j \models_{[T \leftarrow \tau_j]} \psi^{\tau_j - \tau_i}(T)$, that is, $\rho^j \models_{[T, x \leftarrow \tau_j]} \psi^{\tau_j - \tau_i}(x)$, for all j $\geq i$. By the change-of-initial-time lemma it follows that $\rho^j \models_{[T \leftarrow \tau_i, x \leftarrow \tau_j]} \psi(x)$ for all j $\geq$ i, thus letting us infer that $\rho^i \models_{[T \leftarrow \tau_i]} \Box x.$ \$(I). In the case of eventualities the proof proceeds similarly.

(b) Let $\rho = (\sigma, \tau)$ be a K-bounded model of ϕ; the subsets Φ_i, i $\geq$ 0, of $TCl(\phi)$ are defined as follows: $CLOCK_{\tau_{i+1} - \tau_i} \in \Phi_i$ and, for all $\psi \in Cl(\phi)$, $\psi \in \Phi_i$ iff $\rho^i \models_{[T \leftarrow \tau_i]} \psi$. We show that $\Phi = \Phi_0\Phi_1 \ldots$ is an #-path through $\mathcal{T}(\phi)$.

By inspecting the consistency rules, it is evident that every Φ_i is (maximally) consistent. To prove that Φ is an infinite path through $\mathcal{T}(\phi)$, we also have to show that there is an edge from Φ_i to Φ_{i+1} for all $i \geq 0$.

Suppose that $\bigcirc x.\ \psi(x) \in \Phi_i$; that is, $\rho^i \models_{[T \leftarrow \tau_i]} \bigcirc x.\ \psi(x)$ and $\rho^{i+1} \models_{[T \leftarrow \tau_i, x \leftarrow \tau_{i+1}]} \psi(x)$. Let $t = \tau_{i+1} - \tau_i$; then $\rho^{i+1} \models_{[T \leftarrow \tau_i, x \leftarrow \tau_i + t]} \psi(x)$. By the *change-of-initial-time* lemma, $\rho^{i+1} \models_{[T, x \leftarrow \tau_i + t]} \psi^t(x)$, and therefore $\rho^{i+1} \models_{[T \leftarrow \tau_i + t]} \psi^t(T)$. Since also $CLOCK_t \in \Phi_i$, the initial tableau for ϕ contains an edge from Φ_i to Φ_{i+1}.

We now show that the infinite path Φ is indeed a. &path. It satisfies the *progress* condition because τ does. To see that $\phi \in \Phi_0$, observe that ρ is a model of ϕ. It remains to establish that all eventualities in Φ are satisfied "in time."

Suppose that Ox. $\psi(x) \in \Phi_i$; that is, $\rho^i \models_{[T \leftarrow \tau_i]} \Diamond x.\ \psi(x)$ and $\rho^j \models_{[T \leftarrow \tau_i, x \leftarrow \tau_j]} \psi(x)$ for some $j \geq i$. Let $t = \tau_j - \tau_i$; thus $t = \Sigma_{i \leq k < j} Clock(\Phi_k)$. Then $\rho^j \models_{[T \leftarrow \tau_i, x \leftarrow \tau_i + t]} \psi(x)$, and, by the *change-of-initial-time* lemma, $\rho^j \models_{[T, x \leftarrow \tau_i + t]} \psi^t(x)$. Hence $\rho^j \models_{[T \leftarrow \tau_j]} \psi^t(T)$ and $\psi^t(T) \in \Phi_j$. ■

The usual techniques for checking whether a tableau contains an infinite path along which all eventualities are satisfied, can be straightforwardly adopted to check whether the initial tableau contains a ϕ-path: first mark all eventualities that are trivially satisfied, and then repeatedly mark the eventualities that are satisfied in successor states. This procedure is polynomial in the size of the initial tableau, which contains $O(K \cdot 2^{nk})$ states, each of size $O(nk)$.

Thus, $\mathcal{T}(\phi)$ can be constructed and checked for ϕ-paths in deterministic time esponential in nk.

3.1.5 Bounding the time stepwidth

Given a TPTL-formula ϕ, we finally determine a bound Ii on the time increase between two successive states, such that the satisfiability of ϕ is not affected. Let c be the largest constant in ϕ that occurs in a subformula of the form $x \leq S^{c-1}y$ or $S^{c-1}x \leq y$, and $\equiv_{c_1}, \ldots \equiv_{c_m}$ all the congruence predicates occurring in ϕ.

If the time increase t between two states is greater than or equal to c, it obviously suffices to know the residues of t modulo $c_1, \ldots c_m$ in order to update, in a tableau, all timing constraints correctly. Indeed, for checking the satisfiability of ϕ, the arbitrary stepwidth t can be bounded by taking the smallest representative for each of the finitely many congruence classes.

LEMMA **[Bounded time increase].** *If ϕ is satisfiable, then $\rho \models \phi$ for* *some $\rho = (\sigma, \tau)$ with $\tau_{i+1} \leq \tau_i + k$ for all $i \geq 0$, where k is the product of all constants occurring in ϕ.* ■

PROOF. We can, in fact, derive the tighter bound $c + k' \leq k$, for the least common multiple k' of all c_i, $1 \leq i \leq m$. Given a model $\rho = (\sigma, \tau)$ of ϕ, let the time sequence τ' be such that, for all $i \geq 0$, $\tau_{i+1} = \tau'_i + (\tau_{i+1} - \tau_i)$ if $\tau_{i+1} - \tau_i < c$: else choose τ'_{i+1} to be the smallest $t \geq \tau'_i + c$ with $t \equiv_{k'} \tau_{i+1}$. It is easy to see that $\rho' = (\sigma, \tau')$ is also a model of ϕ. ■

Combining this result with the tableau method developed above, we arrive at the following conclusion.

THEOREM [Deciding TPTL]. *The satisfiability* of a TPTL-formula *ϕ is decidable in deterministic time exponential in nk, where n-1 is the number of (propositional and temporal) connectives in ϕ and k is the product of all constants occurring in ϕ.* ■

Note that the length l of a formula whose constants are represented in binary, is $O(n + \log k)$. So we have a decision procedure for TPTL that is doubly exponential in l (although only singly exponential in n, the "untimed" part, and thus, singly exponential for PTL).

The algorithm outlined here can, of course, be improved along the lines of [Wo83]. In particular, we may avoid the construction of the entire initial tableau by starting with the initial state, containing ϕ, and successively adding new states only when needed. This does not, however, lower the doubly exponential deterministic-time bound; in fact, as we show in the following subsection, TPTL is EXPSPACE-hard.

3.2 Complexity of TPTL

THEOREM [Complexity of TPTL]. *Satisfiability for* TPTL is EXPSPACE-complete *(with* respect *to* polynomial time reduction). ■

The proof proceeds in two parts; we first show that TPTL is in EXPSPACE, and then that it is EXPSPACE-hard. The first part follows the argument that PTL is in PSPACE, which builds on a nondeterministic version of the tableau decision procedure ([Wo83]); the hardness part is patterned after the proof of [HU79] that the universality problem of regular expressions with exponentiation is EXPSPACE-hard.

PROOF. [EXPSPACE] We show that satisfiability for TPTL is in nondeterministic EXPSPACE, and hence, by Savitch's theorem, in (deterministic)

EXPSPACE. In particular, it can be checked in nondeterministic singly exponen tial space whet her $\mathcal{T}(\phi)$ contains a \$-path of the form stated in the *length-of-&paths* lemma.

In trying to construct such a ϕ-path nondeterministically, at each stage only the current state, the "loop-back" state, and a state counter have to be retained in order to construct a successor state, loop back, or, if the state counter exceeds the maximal length of the loop, fail. Since both the size of each state and the length of the loop have, by the size-of-closure and *length-of-q&paths* lemmas, respectively, (singly) exponential representations in the length of ϕ, it follows that this nondeterministic decision procedure requires only exponential space.

[EXPSPACE-hardness] Consider a (deterministic) 2^n-**space-bounded** Turing machine M; for each input X of length n, we construct a TPTL-formula ϕ_X of length $O(n \cdot \log n)$ that is satisfiable iff M accepts X. By a standard complexity-theoretic argument, using the hierarchy theorem for space, it follows that there is a constant c > 0 such that every Turing machine solving the satisfiability problem for TPTL-formulas ϕ of length l takes space $S(l) \geq 2^{cl/\log l}$ infinitely often.

Thus it suffices to show, given X, how to construct a sufficiently succinct formula ϕ_X that describes the (unique) computation of M on X, as an infinite sequence of propositions, and requires it to be accepting. We use a proposition p_i and a proposition q_j for every tape symbol i and state j of M, respectively. In particular, p_0 and q_0 correspond to the special tape symbol "blank" and the initial state of M. Let

$$\hat{p}_i = p_i \wedge \bigwedge\nolimits_{i' \neq i} \neg p_{i'} \wedge \bigwedge \neg q_j,$$
$$r_{i,j} = \mathrm{Pi} \wedge q_j \wedge \bigwedge\nolimits_{i' \neq i} \neg p_{i'} \;\mathrm{A}\; \bigwedge\nolimits_{j' \neq j} \neg q_{j'},$$
$$s = \bigwedge \neg p_{i'} \;\mathrm{A}\; \bigwedge \neg q_j.$$

We represent configurations of M by $\hat{p}$-**state** sequences of length 2^n, which are separated by s-states; the position of the read-write head is marked by an r-state. The computation of M on X is completely determined by the following two conditions:

(i) it starts with the initial configuration, and
(ii) every configuration follows from the previous one by a move of M.

The computation is accepting iff, furthermore,

(*iii*) it contains the accepting state F.

These conditions can be expressed in TPTL; take ϕ_X to consist of $\Box x. \bigcirc y. y = x + 1$, forcing time to resemble a state counter, and the following three conjuncts, corresponding to (i)–(iii):

$$\phi_{INITIAL} = \left(\begin{array}{l} s \wedge \bigcirc r_{X_1,0} \wedge \\ \bigwedge_{2 \leq i \leq n} \Box x. (x = i \rightarrow \hat{p}_{X_i}) \wedge \\ \Box x. (n < x \leq 2^n \rightarrow \hat{p}_0) \end{array} \right)$$

$$\phi_{MOVE} = \left(\begin{array}{l} \Box x. (s \rightarrow \Diamond y. (y = x + 2^n + 1 \wedge s)) \wedge \\ \bigwedge_{P,Q,R} \Box x. \left(\begin{array}{l} P \wedge \bigcirc Q \wedge \bigcirc^2 R \rightarrow \\ \Diamond y. (y = x + 2^n + 2 \wedge f_M(P, Q, R) \end{array} \right) \end{array} \right)$$

$$\phi_{ACCEPT} = \Diamond \bigvee r_{i,F}$$

(P, Q, and R each range over the propositions $\hat{p}_i$, $r_{i,j}$, and s, and $f_M(P, Q, R)$ refers to the transition function of M. For instance, $f_M(\hat{p}_i, r_{i',j}, \hat{p}_{i''}) = \hat{p}_k$ and $f_M(r_{i',j}, \hat{p}_{i''}, \hat{p}_{i'''}) = r_{i'',j'}$ if M writes, in state j on input i', the symbol k onto the tape, moves to the right, and enters state j'.)

The lengths of $\phi_{INITIAL}$, ϕ_{MOVE}, and ϕ_{ACCEPT} are $O(n \cdot log\, n)$, $O(n)$, and O(1), respectively (recall that constants are represented in binary), thus implying the desired $O(n \cdot \log n)$-bound for ϕ_X. ∎

3.3 Real-time verification

We define finite-state real-time systems, and give an algorithm that checks whether such a system satisfies a TPTL-specification.

This problem, of checking whether a TPTL-formula ϕ is satisfied in a given structure ("model checking"), is again EXPSPACE-complete, and thus, in general, no simpler than determining if ϕ is satisfiable at all. Its complexity is, however, doubly exponential only in the size of ϕ, which is usually much smaller than the size of the structure (on which the dependence is singly exponential).

3.3.1 Real-time systems

A program may be abstractly viewed as a state-transition graph ([MP89]). We model real-time systems by such graphs each of whose transitions is labeled by a subset of TIME, denoting the nondeterministic amount of time that transition consumes.

DEFINITION [Syntax]. Let P be a finite set of propositions. A *real-time system* $\mathcal{S} = (S, T, \sigma_0)$ over P is a finite. directed graph whose vertices S are marked by subsets of P and whose edges $T \in S^2 \times 2^{TIME}$ are labeled by subsets of $TIME$. The vertices and edges are called the *states* and the *transitions* of S, respectively; $\sigma_0 \in S$ is the *initial state* of S.

The real-time system $\mathcal{S}$ is *finite-state* iff all transitions $\sigma \xrightarrow{I} \sigma'$ are labeled by (finite or infinite) time intervals $I = [t_1, t_2]$, where $t_1 \in TIME$ and $t_2 \in TIME \cup \{\infty\}$. ■

In accordance with the intuitive operational semantics of a real-time system S, we define its denotational trace semantics $[\![\mathcal{S}]\!]$ to be a set of timed state sequences, the runs of S. We say that S *satisfies* the TPTL-formula ϕ iff some sequence in $[\![\mathcal{S}]\!]$ is a model of ϕ.

DEFINITION [Semantics]. For any real-time system S = (S, T, σ_0), the timed state sequence $\rho = (\sigma, \tau)$ belongs to $[\![\mathcal{S}]\!]$ iff there is an infinite path $\sigma_0 \xrightarrow{I_0} \sigma_1 \xrightarrow{I_1} \sigma_2 \xrightarrow{I_2} \ldots$ through S such that $\tau_{i+1} - \tau_i \in I_i$ for all $i \geq 0$. ■

The problem of model checking is to determine whether a finite-state real-time system satisfies a TPTL-formula. Model-checking algorithms can be used to verify the property ψ of the system S: in order to show that ψ holds over all runs of S, it suffices to check that $[\![\mathcal{S}]\!]$ does not contain a model of $\neg\psi$.

3.3.2 Model checking

By associating a tableau $\mathcal{T}(\mathcal{S})$ with a given finite-state real-time system $\mathcal{S}$, and applying our TPTL-tableau techniques, we may adopt the model-checking algorithms for PTL ([LP84]). We only sketch the method; its basic idea is that, in order to check whether $\mathcal{S}$ satisfies ϕ, we form the product of both tableaux, $\mathcal{T}(\mathcal{S})$ and $\mathcal{T}(\phi)$, and check the resulting tableau for &paths.

Let $CLOCK_{t_1,t_2}$, $t_1 \in TIME$ and $t_2 \in TIME \cup \{\infty\}$, be new propositions, representing the finite and infinite time intervals $[t_1, t_2]$. Accordingly, we add the following rule to the definition of the (local) *consistency* of a set Φ of **TPTL-formulas:**

- **If** $CLOCK_{t_1,t_2}$ is in Φ, then $t_1 \leq Clock(\Phi) \leq t_2$ (where $t < \infty$ for all $t \in TIME$).

Using these interval propositions instead of edge labels, we obtain, from the finite-state real-time system $\mathcal{S}$ over P, the following tableau $\mathcal{T}(\mathcal{S})$ over $P \cup \{ CLOCK_{t_1,t_2} \mid t_1 \leq t_2\}$.

Every state σ of $\mathcal{S}$ is split into a finite set $\mathcal{T}(\sigma)$ of states, one for each outgoing transition, while preserving all incoming transitions. If a state has an outgoing transition labeled by $[t_1, t_2]$, the corresponding proposition $CLOCK_{t_1,t_2}$ is added to the state: all transition labels may then be deleted. Since any state of $\mathcal{S}$ has at most one edge to any other state, the size of $\mathcal{T}(\mathcal{S})$ is polynomial in the size of S.

Let us write $\mathcal{T} = (S, T)$ for the timed tableau with vertex set S and edge set T. The *product* $\mathcal{T}_1 \times \mathcal{T}_2$ of two tableaux $\mathcal{T}_1 = (S_1, T_1)$ and $\mathcal{T}_2 = (S_2, T_2)$ is the directed graph whose vertices are the consistent sets $\sigma_1 \cup \sigma_2$, for $\sigma_1 \in S_1$ and $\sigma_2 \in S_2$, and which contains an edge from $\sigma_1 \cup \sigma_2$ to $\sigma'_1 \cup \sigma'_2$ iff both $(\sigma_1, \sigma'_1) \in T_1$ and $(\sigma_2, \sigma'_2) \in T_2$. The size of (the tableau) $\mathcal{T}_1 \times \mathcal{T}_2$ is clearly linear in the product of the sizes of $\mathcal{T}_1$ and $\mathcal{T}_2$.

We obtain the following result immediately from the definitions, which gives us an algorithm for model checking.

LEMMA [Correctness of tableau product]. *For any finite-state real-time system $\mathcal{S} = (S, T, \sigma_0)$ and* TPTL-formula ϕ, *$\mathcal{S}$ satisfies ϕ iff $\mathcal{T}(\mathcal{S}) \times \mathcal{T}(\phi)$ contains a ϕ-path whose first projection starts in $\mathcal{T}(\sigma_0)$.* ∎

For the previous lemma to go through, the bound K for $\mathcal{T}(\phi)$ is chosen to be greater than the largest constant $t < \infty$ occurring in S. Thus, the size of $\mathcal{T}(\phi)$, which contains $O(K^l \cdot 2^{2^l})$ states, and therefore the running-time of the model-checking algorithm, depend singly exponentially on the size of $\mathcal{S}$ and doubly exponentially on the length l of ϕ.

According to different versions of fairness, various variants of the notion of ϕ-paths can be defined, and checked for, as in [LP84].

3.3.3 Complexity of model checking

THEOREM [Complexity of model checking]. *Determining whether a finite-state real-time system satisfies a* TPTL-formula *is* EXPSPACE-*complete.* ∎

Model checking is polynomial-time reducible to the satisfiability problem for TPTL; the proof follows [SC85]. The converse holds trivially.

PROOF. [EXPSPACE] Given the formula ϕ and the finite-state real-time system $\mathcal{S} = (S, T, \sigma_0)$, we show how to construct a TPTL-formula $\phi_{\mathcal{S}}$, whose length depends polynomially on the sizes of both $\mathcal{S}$ and ϕ, and which is satisfiable iff $\mathcal{S}$ satisfies ϕ at σ_0.

For every state $\sigma_i \in S$, we introduce a new proposition, p_i, and use the abbreviation

$$\psi_i = p_i \rightarrow \Big(\bigwedge_{q\in\sigma_i} q \wedge \bigwedge_{q\notin\sigma_i} \neg q \wedge \bigvee_{(\sigma_i,\sigma_j,[t_1,t_2])\in T} \bigcirc y.(p_j \wedge x+t_1 \leq y \leq x+t_2)\Big).$$

Furthermore. let ψ assert that exactly one of the propositions p_i, $\sigma_i \in S$, is true. It is easy to see that the formula

$$\phi_S = \phi \wedge p_0 \wedge \Box x.(\psi \wedge \bigwedge_{\sigma_i\in S} \psi_i)$$

has the desired properties.

[EXPSPACE-hardness] To reduce the satisfiability problem for TPTL to model checking, it suffices to give a finite-state real-time system S of constant size such that S satisfies the TPTL-formula ϕ iff ϕ is satisfiable. Simply choose S to be the complete graph over all subsets of P, the propositions occurring in ϕ, and label all edges by $[0,\infty]$. ▪

4 Undecidable Extensions of TPTL

We consider two natural extensions of TPTL, a syntactic one (allowing addition over time) and a semantic one (interpreting TPTL-formulas over a dense time domain). Both extensions are shown to be Π^1_1-complete, by reducing a Σ^1_1-hard problem of 2-counter machines to the respective satisfiability problems. It follows that they cannot even be (recursively) axiomatized (for an exposition of the analytical hierarchy consult, for instance, [Ro67]).

4.1 A Σ^1_1-complete problem

A *nondeterministic 2-counter machine* M consists of two counters C and D, and a sequence of n instructions, each of which may increment or decrement one of the counters, or jump, conditionally upon one of the counters being zero. After the execution of a non-jump instruction, M proceeds nondeterministically to one of two specified instructions.

We represent the configurations of M by triples (i, c, d), where $0 \leq i < n$, $c \geq 0$, and $d \geq 0$ are the current values of the location counter and the two counters C and D, respectively. The consecution relation on configurations is defined in the obvious way. A *computation* of M is an infinite sequence of related configurations, starting with the initial configuration $\langle 0,0,0\rangle$. It is called *recurring* iff it contains infinitely many configurations with the value of the location counter being 0.

The problem of deciding whether a nondeterministic Turing machine has, over the empty tape, a computation in which the starting state is visited infinitely often, is known to be Σ_1^1-complete ([HPS83]). Along the same lines we obtain the following result.

LEMMA [Complexity of 2-**counter** machines]. *The problem of deciding whether a given nondeterministic 2-counter machine has a recurring computation, is Σ_1^1-hard.* ▪

PROOF. Every Σ_1^1-formula is equivalent to a Σ_1^1-formula χ of the form $\exists f.\,(f(0) = 1 \land \forall x.\, g(f(x), f(x+1)))$, for a recursive predicate g ([HPS83]). For any such χ we can construct a nondeterministic 2-counter machine M that has a recurring computation iff χ is true.

Let M start by computing f(0) = **1,** and proceed, indefinitely, by nondeterministically guessing the next value of f. At each stage, M checks whether f(x) and $f(x+1)$ satisfy g, and if (and only if) so, it jumps to instruction 0. Such an M exists, because 2-counter machines can, being universal, compute the recursive predicate g. It executes the instruction 0 infinitely often iff a function f with the desired properties exists. ▪

4.2 Encoding computations of Z-counter machines

We show that the satisfiability problem for several extensions of TPTL is Σ_1^1-complete.

First, we observe that the satisfiability of a formula ϕ can, in all cases, be phrased as a Xi-sentence, asserting the existence of a model for ϕ. Any timed state sequence ρ for ϕ can be encoded, in first-order arithmetic, by finitely many infinite sets of natural numbers; say, one for each proposition p in ϕ, characterizing the states in which p holds, and one to encode state-time pairs. It is routine to express, as a first-order predicate, that ϕ holds in ρ. We conclude that satisfiability is in Σ_1^1.

To show that the satisfiability problem of a logic is Cf-hard, it suffices, given a nondeterministic 2-counter machine M, to construct a formula ϕ_M such that ϕ_M is satisfiable iff M has a recurring computation. We demonstrate the technique of encoding recurring computations of M by showing that the *monotonicity* constraint on time is necessary for the decidability of TPTL.

THEOREM [Nonmonotonic time]. *Relaxing the monotonicity condition for time sequences renders the satisfiability problem for* TPTL Σ_1^1-*complete.* ▪

PROOF. We encode the computation I' of M by the time sequence τ such that, for all $k \geq 0$, $\tau_{3k} = i$, $\tau_{3k+1} = n + c$, and $\tau_{3k+2} = n + d$ for the k-th configuration $\langle i$, c, $d\rangle$ of Γ. Now it is easy to express, by a TPTL-formula ϕ_M, that a time sequence encodes a recurring computation of M.

First specify the initial configuration, by

$$\phi_{INITIAL} = \mathbf{(T = 0\ A} \bigcirc x.\, x = \mathbf{0\ A} \bigcirc^2 x.\, x = \mathbf{0);}$$

then ensure proper consecution by adding a O-conjunct ϕ_i for every instruction i of M. For instance, the instruction 1 that increments the counter C and proceeds, nondeterministically, to either instruction 2 or 3, contributes the conjunct

$$\phi_1 = \mathbf{ax.} \begin{pmatrix} \mathbf{x = 1} \rightarrow \bigcirc^3 y.\, \mathbf{(y = 2} \vee \mathbf{y = 3)} \wedge \\ \bigcirc y.\, \bigcirc^3 z.\, z = y + 1 \wedge \\ \bigcirc^2 y.\, \bigcirc^3 z.\, z = \mathrm{y} \end{pmatrix}$$

The recurrence condition can be expressed by a □◇-formula,

$$\phi_{RECUR} = \mathbf{002.X = 0.}$$

Clearly, the conjunction ϕ_M of these $n + 2$ formulas is satisfiable iff M has a recurring computation. •

Note that we do not require any propositions in the proof. It follows that first-order temporal logic with a single state variable ranging over the natural numbers is Π_1^1-complete, provided the underlying assertion language has at least successor (in addition to equality) as a primitive.

4.3 Presburger TPTL

We show that a certain extremely modest relaxation of the syntax of timing **constraints** leads to **a** highly undecidable logic. Consequently, TPTL with addition over time is undecidable.

THEOREM [Presburger TPTL]. *If* the *syntax* of TPTL is *extended to allow multiplication by 2, the satisfiability problem becomes Xi-complete.* ▪

PROOF. To encode computations of M, we use the propositions $p_1, \ldots p,,$ r_1, and r_2, precisely one of which is true in any state; hence we may identify states with propositions. The configuration (i, c, $d\rangle$ of M is represented by the finite sequence $p_i\ r_1^c\ r_2^d$ of states.

The initial configuration (p_0) as well as the recurrence condition $(\Box\Diamond p_0)$ can be easily expressed in PTL. The crucial property that allows a temporal logic to specify the consecution relation of configurations, and thus the set of computations of M, is the ability to copy an arbitrary number of r-states. In timed temporal logics, the times associated with a state sequence can be used for copying.

With the availability of multiplication by 2, we are able to have the k-th configuration of a computation correspond, for all $k \geq 0$, to the finite sequence of states that is mapped to the time interval $[2^k, 2^{k+1})$. First, we force the time to increase by a strictly positive amount between successive states (Ox. $\bigcirc$ y. y > x), to ensure that every state is uniquely identifiable by its time. Then we can copy groups of r-states by establishing a one-to-one correspondence of r_j-states $(j = 1, 2)$ at time t and time $2t$; clearly there are enough time gaps to accommodate an additional r_j-state when required by an increment instruction.

For instance, the instruction 1 that increments the counter C and proceeds, nondeterministically, to either instruction 2 or 3, can be expressed as follows:

$$\text{OX.} \left(\begin{array}{l} p_1 \rightarrow \Diamond z.\ (z = 2x \wedge (p_2 \vee p_3)) \wedge \\ \Box y_1.\bigcirc y_2.(y_2 < 2x \rightarrow \Diamond z_1.\ (z_1 = 2y_1\ A\ \bigcirc z_2.\ z_2 = 2y_2)) \wedge \\ \bigwedge_{i=1,2} \Box y.\ (y < 2x\ \mathsf{A}\ r_i \rightarrow \text{OZ.}\ (z = 2y\ \mathsf{A}\ r_i)) \wedge \\ \Box\ [illegible]\ \bigcirc\ y_2.(y_2 = 2x \rightarrow \Diamond z_1.(z_1 = 2y_1\,\mathsf{A}\bigcirc z_2.r_1\,\mathsf{A}\bigcirc^2 z_3.z_3 = 2y_2)) \end{array} \right)$$

The first conjunct ensures the proper progression to one of the two specified instructions, 2 or 3; the second one establishes a one-to-one correspondence between states in successive intervals representing configurations, while the third and fourth conjuncts copy r_j-states $(j = 1, 2)$. The last conjunct adds, finally, an r_1-state at the end of the successor configuration, as required by the increment operation. •

We can modify this proof by reducing time to a state counter (Ox. $\bigcirc\, y.y = x + 1)$, and letting all propositions be false in the resulting additional (padding) states. Thus, the satisfiability problem for TPTL with multiplication by 2 is Σ_1^1-hard even if time is replaced by a state counter. As a corollary we infer that the first-order theory of the natural numbers with $\leq$, multiplication by 2, and monadic predicates is Π_1^1-complete. A similar result has been obtained by [AH89].

4.4 **Dense TPTL**

Another possible direction to extend the expressive power of TPTL is to relax its semantics by adopting a dense time domain (i.e., between any two given time points there is another time point). We show that the resulting logic is, again, highly undecidable.

THEOREM [Dense TPTL]. *If* TPTL *is interpreted over the rationals (i.e., TIME = Q), the satisfiability problem becomes Xi-complete.* ∎

PROOF. The proof depends, once more, on the ability to copy groups of r-states. This time, we are able to have the k-th configuration of a computation of M correspond, for all $k \geq 0$, to the finite sequence of states that is mapped to the time interval [k, $k + 1$), because dense time allows us to squeeze arbitrarily many states into every interval of length 1.

Again, we identify every state with a unique time, and can then establish a one-to-one correspondence of r_j-states ($j = 1, 2$) at time t and time $t + 1$. In fact, we may simply replace all occurrences of multiplication by 2 in the Presburger-TPTL formula encoding the recurring computations of M, by a successor operation, in order to obtain the desired dense-TPTL formula ϕ_M. ∎

This proof goes through for any time domain *(TIME,* $<$, S) such that (*TIME,* <) is a dense linear order, and S is a unary function **over** *TIME* satisfying the first-order axioms $\forall x.$ x < $S(x)$ and $\forall x, y.$ (a: < $y \rightarrow S(x)$ < S(y)). To show that, for arbitrary dense time domains, the satisfiability problem is in Σ_1^1 , a standard Löwenheim-Skolem argument is necessary to infer the existence of countable models.

The proof technique outlined here can, in fact, be applied to many other real-time logics, such as the logic of [KVD83], RTL ([JM86]), RTTL ([Os87]), and GCTL ([Ha88]). All of these formalisms admit addition over time as a primitive, which renders them undecidable.

This suggests that we have been able to characterize an intrinsic boundary between the decidability and undecidability of formalisms that combine **finite-state** reasoning about state sequences with explicit reasoning about time. Such logics **are** undecidable if they permit Presburger arithmetic on time, as well as if they are interpreted over dense models of time.

5 Discussion

We have demonstrated a very natural way to extend qualitative temporal reasoning over state sequences to quantitative temporal reasoning over timed state sequences. We then identified the restrictions on syntax and semantics necessary for achieving elementary decidability. The language TPTL is, so we believe, a good real-time specification language, and its model-checking algorithm can be used for the verification and synthesis of real-time systems, along the lines of [LP84] and [MW84].

5.1 Some comparisons with related work

Several researchers have proposed to add real-time modalities like $\Diamond_{\leq k}$ ("eventually within time *k*") to PTL ([KVD83], [Ha88], [EMSS89]) — a notation that is easily seen to be subsumed by TPTL: $\Diamond_{<k}\ \phi$ is equivalent to Ox. $(x \leq k$ A $\phi)$. Others have argued for identifying "next-state" with "next-time" ([EMSS89]). This approach is again a special case of ours (Ox. $\bigcirc$ y. y = x + 1), and fails for asynchronous systems, whose events can occur arbitrarily close in time.

The solution of adding explicit time independent of the state-transition relation, and thus admitting multiple successive states with the same time, is usually pursued by adopting a first-order temporal logic, with a dynamic time variable. For RTTL ([Os87]), the questions of appropriate quantification and decidability have not been addressed. **For GCTL** ([Ha88]), it has been shown that the satisfiability of the existential closure of quantifier-free formulas is decidable. This result restricts the language to only one form of quantification, and even then cannot be used for verification (i.e., proving that an implementation implies a specification).

5.2 Some directions for future work

We hope that the language TPTL will generate sufficient interest to warrant the **study of its applicability** to the **specification,** verification, and synthesis of real-time **systems.** Of particular importance is the development of a suitable, compositional modeling that yields to the tableau methods presented in this **paper.**

We also plan to investigate the expressive power **of** TPTL, as well as the consequences of extending TPTL to timed ETL ([Wo83]) and of introducing *past* temporal quantifiers ([LPZ85]). The classification of "really temporal" properties into a hierarchy similar to the conventional safety-liveness distinction of PTL ([MP89]), and the addition of temporal quantification to branching-time logics ([EC82]) remain to be studied.

Perhaps most importantly, the consequences of restricting arbitrary quantifiers in first-order modal logics, to *modal quantifiers* that are associated with modal operators, has to be pursued independently of the notion of time.

Acknowledgements. We thank Zohar Manna, Amir Pnueli, and David Dill for their guidance. Moshe Vardi and Joe Halpern gave us very helpful advice for refining our undecidability results; in particular, they pointed out to us the Σ_1^1-completeness of a problem on Turing machines.

References

[AH89] M. Abadi, J. Halpern, "Decidability and expressiveness for first-order logics of probability," 30th IEEE FOCS, 1989.

[BH81] A. Bernstein, P.K. Harter, "Proving real-time properties of programs with temporal logic," 8th ACM Symp. on Operating System Principles, 1981.

[EC82] E.A. Emerson, E.C. Clarke, "Using branching time temporal logic to synthesize synchronization skeletons," Science *of* Computer Pro*gramming 2, 1982.*

[EMSS89] E.A. Emerson, A.K. Mok, A.P. Sistla, J. Srinivasan, "Quantitative temporal reasoning," presented at the Workshop on Finite-state Concurrency, Grenoble, France, 1989.

[Ha88] E. Harel, *Temporal Analysis of Real-time Systems,* M.S. Thesis, Weizmann Institute, 1988.

[HPS83] D. Harel, A. Pnueli, J. Stavi, "Propositional dynamic logic of regular programs," *J. computer and System Sciences 26*, 1983.

[HU79] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[JM86] F . Jaanian, A.K. Mok, "Safety analysis of timing properties in real-time systems," IEEE Trans. on Soft ware Engineering SE-12, 1986.

[KVD83] R. Koymans, J. Vytopil, W.P. de Roever, "Real-time programming and asynchronous message passing," 10th ACM POPL, 1983.

[LP84] 0. Lichtenstein, A. Pnueli, "Checking that finite-state concurrent programs satisfy their linear specification," 11th ACM POPL, 1984.

[LPZ85] 0. Lichtenstein, A. Pnueli, L. Zuck, "The glory of the past," Conf. on Logics of Programs, Springer *LNCS* **193**, 1985.

[MP89] Z. Manna, A. Pnueli, "The anchored version of the temporal framework," *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency* (J.W. deBakker, W.-P. de Roever, and G. Rozenberg, eds.), Springer *LNCS* **354**, 1989.

[MW84] Z. Manna, P. Wolper, "Synthesis of communicating processes from temporal logic specifications," ACM *TOPLAS* **6**, 1984.

[NA88] K.T. Narayana, A.A. Aaby, "Specification of real-time systems in real-time temporal interval logic," IEEE Real-time Systems Symp., 1988.

[OL82] S. Owicki, L. Lamport, "Proving liveness properties of concurrent programs," ACM *TOPLAS 4*, 1982.

[Os87] J.S. Ostroff, *Temporal Logic of Real-time Systems,* Ph.D. Thesis, Univ. of Toronto, 1987 (to be published by Research Studies Press).

[PH88] A. Pnueli, E. Harel, "Applications of temporal logic to the specification of real-time systems," Formal Techniques in Real-time and Fault-tolerant Systems, Springer *LNCS* **331**, 1988.

[Pn77] A. Pnueli, "The temporal logic of programs," 18th IEEE FOCS, 1977.

[Ro67] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability,* McGraw-Hill, 1967.

[SC85] A.P. Sistla, E.M. Clarke, "The complexity of propositional linear temporal logics," *JACM* **32**, 1985.

[Wo83] P. Wolper, "Temporal logic can be more expressive," Information *and* Control **56**, *1983.*