From Quantity to Quality[⋆,⋆⋆]

Thomas A. Henzinger Orna Kupferman

UC Berkeley, EECS Department, Berkeley, CA 94720-1770, U.S.A.
Email: {tah,orna}@eecs.berkeley.edu

Abstract. In temporal-logic model checking, we verify the correctness of a program with respect to a desired behavior by checking whether a structure that models the program satisfies a temporal-logic formula that specifies the behavior. The model-checking problem for the branching-time temporal logic CTL can be solved in linear running time, and model-checking tools for CTL are used successfully in industrial applications. The development of programs that must meet rigid real-time constraints has brought with it a need for real-time temporal logics that enable quantitative reference to time. Early research on real-time temporal logics uses the discrete domain of the integers to model time. Present research on real-time temporal logics focuses on continuous time and uses the dense domain of the reals to model time. There, model checking becomes significantly more complicated. For example, the model-checking problem for TCTL, a continuous-time extension of the logic CTL, is PSPACE-complete.

In this paper we suggest a reduction from TCTL model checking to CTL model checking. The contribution of such a reduction is twofold. Theoretically, while it has long been known that model-checking methods for untimed temporal logics can be extended quite easily to handle discrete time, it was not clear whether and how untimed methods can handle the reset quantifier of TCTL, which resets a real-valued clock. Practically, our reduction enables anyone who has a tool for CTL model checking to use it for TCTL model checking. The TCTL model-checking algorithm that follows from our reduction is in PSPACE, matching the known bound for this problem. In addition, it enjoys the wide distribution of CTL model-checking tools and the extensive and fruitful research efforts and heuristics that have been put into these tools.

1 Introduction

Temporal logics can describe a temporal ordering of events and have been adopted as a powerful tool for specifying and verifying concurrent programs [Pnu77, MP92]. In temporal-logic *model checking*, we verify that a program meets a desired behavior by checking that a mathematical model of the program satisfies a temporal-logic formula that specifies the behavior [CE81, QS81]. We distinguish between four levels of temporal reasoning. The verification methods induced by these levels differ in the interpretation given to time.

⋆ A preliminary version of this paper appeared in the *Proceedings of the First International Workshop on Hybrid and Real-time Systems* (HART 97), *Lecture Notes in Computer Science* **1201**, Springer-Verlag, 1997, pp. 48–62.

⋆⋆ This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

The first level allows only qualitative reference to time. The classical method of CTL model checking belongs to this level. There, the program is modeled as a state-transition graph, and the correct behaviors of the program are specified in the qualitative branching temporal logic CTL, which allows temporal operators such as "always" and "eventually." For example, if *req* and *grant* are atomic propositions, then the CTL formula

$$\psi = AG(req \rightarrow AF\, grant)$$

asserts that in all computations of the program, every request is eventually granted. The model-checking problem for CTL can be solved in linear time. More precisely, given a state-transition graph K and a CTL formula ψ, we can determine whether K satisfies ψ in time $O(|K| \cdot |\psi|)$ [CES86].

The development of programs that must meet rigid *real-time* constraints has brought with it a need for *real-time temporal logics* that enable quantitative reference to time [EMSS90, AH92]. We consider here a real-time extension of CTL, which we call CTL+clocks. The syntax of CTL+clocks extends the syntax of CTL by allowing reference to a set of *clock variables*. Formulas of CTL+clocks can refer to the values of the clocks, and may contain a *reset quantifier* $c.\varphi$, which resets a clock c to the value 0. For example, if c is a clock, then the formula

$$\psi' = AG[c.req \rightarrow AF(grant \wedge (c \leq 2))]$$

strengthens the formula ψ above by putting an upper bound on the time that may elapse before a grant is given: if the clock c is started at the time of a request, then the value of c is at most 2 at the time of the subsequent grant. The exact meaning of ψ' depends on the formal interpretation of CTL+clocks. The formulas of CTL+clocks can be interpreted in three different ways, forming the following three levels of quantitative temporal reasoning.

Pioneering work on real-time temporal logics allowed very simple quantitative reference to time. In [EMSS90], Emerson et al. interpret CTL+clocks formulas[3] over state-transition graphs (see also [CCM+94]). Each transition in the graph advances the time by one time unit. Hence, this level of temporal reasoning uses the *discrete* domain of the *integers* to model time, and it uses quantitative reference to time only in the specifications. It is quite clear that, when interpreted over state-transition graphs, CTL+clocks formulas can be translated to equivalent CTL formulas, and the problem of CTL+clocks model checking can be reduced to the problem of CTL model checking.[4] For example, when interpreted over state-transition graphs, the formula ψ' above is equivalent to the CTL formula

$$\psi'' = AG(req \rightarrow (grant \vee AX(grant \vee AX\, grant))),$$

which states that every request is granted within two transitions. The main limitation of this level of temporal reasoning is that while discrete time suffices for modeling globally-clocked programs, continuous time is required for modeling the composition of independently-clocked programs.

This limitation has been removed in the third level of temporal reasoning, known as the *fictitious-clock* approach [HMP92, AH93, AH94]. At this level, transitions happen in continuous time, but are recorded by a global digital clock, in discrete time. Accordingly,

[3] The logic used in [EMSS90] is RTCTL, which is a strict syntactic subset of CTL+clocks.

[4] We note that while the translation involves an exponential blow-up, using a more sophisticated approach, model checking can still be done in time linear in the original CTL+clocks formula [EMSS90].

time is viewed as a state variable that ranges over the domain of the integers. Some transitions in the state-transition graph are designated as *tick* transitions (i.e., transitions of the global digital clock). Whenever a tick transition is taken, time is advanced by one time unit. Hence, any number of program transitions can be taken in one time unit. By introducing a new atomic proposition, "*tick*", we can translate CTL+clocks formulas, when interpreted over state-transition graphs with tick transitions, to CTL formulas.[5] For example, the CTL+clocks formula ψ' above is equivalent to the CTL formula

$$\psi''' = AG(req \rightarrow A((\neg tick)U(grant \vee (tick \wedge AXA((\neg tick)U grant))))).$$

Hence, CTL+clocks model checking can be reduced to CTL model checking on this level of temporal reasoning as well [AH92]. The main limitation of this level is its limited accuracy. For example, the formula ψ''' asserts only that in all computations of the program, every request is followed by a grant within more than one and less than three time units. Also, as in the previous level, we have to fix a time granularity, which may cause a blow-up in the state space.

Much present research on qualitative reasoning focuses on *dense* time and uses the domain of the *reals* to model time in both the state-transition graph and the specification. On this fourth level of temporal reasoning, we model the programs as *timed automata* [AD94], where real-valued clocks keep track of timing constraints. The fourth level constitutes the most expressive way of specifying real-time programs. With this semantics, the logic CTL+clocks is called TCTL [ACD93], and the formula ψ' asserts that in all computations of the program, every request is followed by a grant within at most two time units. *TCTL model checking* is the problem of determining whether a given TCTL formula is satisfied in a given timed automaton.

The introduction of dense time in the model makes quantitative reasoning more complicated. For example, while the satisfiability problem for CTL+clocks when interpreted over discrete-time models (levels two or three) is decidable, it is undecidable for TCTL. Indeed, algorithms that handle satisfiability or model checking of CTL, and which are applicable to verification methods induced by the first three levels of temporal reasoning, cannot be easily extended to handle TCTL. The reason is the dense time domain of TCTL, which induces state-transition graphs with infinitely many states. It was shown, however, in [AD94], that each timed automaton induces a finite quotient of the infinite state space, such that two equivalent states satisfy the same TCTL formulas. More precisely, the automaton partitions the infinite time domain of clock valuations into finitely many *regions*, each of which can be viewed as a set of clock constraints (e.g., $2 < clock_1 < 3 \; ; clock_2 = 1$). This finite quotient is used in [ACD93] in order to solve the model-checking problem for TCTL. Alur et al. also prove that the problem is PSPACE-complete. The importance of the model-checking problem has led to the development of several other model-checking algorithms for TCTL [HNSY94, LL95, SS95, HKV96], all trying to cope with the large state space that needs to be stored.

This space problem, known as the *state-explosion problem*, is the main computational limitation of all the verification methods induced by the four levels of temporal reasoning. It constitutes one of the most challenging issues in the area of computer-aided verification and is the subject of active research. Most of the efforts during the last two decades have focused on pure qualitative reasoning, yielding CTL model-checking tools (e.g., SMV, VIS, CADP) that can handle systems with large state spaces [McM93, CGL93, BHSV+96, FGK+96]. Model-checking algorithms for TCTL adopt some of the techniques used in the tools for

[5] Again, the translation involves an exponential blow-up that is unnecessary for model checking.

CTL model checking. Still, TCTL model-checking tools are less successful than CTL model-checking tools, both in their level of performance and in their distribution. The reason is not only the clear computational advantage of CTL, but also the broad attention that CTL model-checking tools have enjoyed.

In this paper we suggest a reduction from TCTL model checking to CTL model checking. The contribution of such a reduction is twofold. Theoretically, it completes the picture of the four levels of temporal reasoning. While it has long been known that the first three levels are inter-reducible, our reduction shows that, as far as model-checking is concerned, the fourth level can also be reduced to the first three levels. Practically, our reduction enables anyone who has a tool for CTL model checking to use it for TCTL model checking. The TCTL model-checking algorithm that follows from our reduction is in PSPACE, matching the known bound for this problem. In addition, it enjoys the extensive and fruitful research efforts that have been put into CTL model-checking tools.

The reduction is not complicated. Given a timed automaton $\mathcal{U}$ and TCTL formula ψ, we construct a state-transition graph $untime(\mathcal{U})$, of size exponential in the size of $\mathcal{U}$, and a CTL formula $untime(\psi)$, of length linear in the length of ψ, such that $\mathcal{U}$ satisfies ψ iff $untime(\mathcal{U})$ satisfies $untime(\psi)$. The graph $untime(\mathcal{U})$ is essentially the region graph used in [ACD93], augmented by a new atomic proposition and new transitions, which handle the reset quantifier $c.\varphi$. When we evaluate TCTL formulas, the reset quantifier causes the course of evaluation to "jump around in the graph." Having to evaluate a formula $c.\varphi$ in a state w, we actually evaluate the formula φ in another state, which differs from w only in that the value of the clock c is reset to 0. Such jumps are replaced in $untime(\mathcal{U})$ by new transitions. In $untime(\psi)$, path quantification guarantees that whenever we come to evaluate $c.\varphi$, the current value of c is 0, and thus no jump is required.

2 Definitions

2.1 Kripke Structures and Timed Structures

We model programs without real-time constraints by *Kripke structures*. A Kripke structure is a tuple $\mathcal{K} = \langle AP, W, R, w^0, \sigma \rangle$, where AP is a finite set of atomic propositions, W is a set of states, $R \subseteq W \times W$ is a total transition relation (every state has at least one successor), $w^0 \in W$ is an initial state, and $\sigma : W \to 2^{AP}$ maps to each state a set of atomic propositions true in the state. A *path* in $\mathcal{K}$ is an infinite sequence of states $w_0, w_1, w_2, \ldots$ such that for all $i \geq 0$, we have $\langle w_i, w_{i+1} \rangle \in R$.

We model real-time programs by *timed structures*. Timed structures extend traditional Kripke structures by labeling each transition with a nonnegative real number denoting its duration. Formally, a timed structure is a tuple $\mathcal{T} = \langle AP, W, R, w^0, \sigma \rangle$, where AP, W, w^0, and σ are as in a Kripke structure, and $R \subseteq W \times \mathbb{R} \times W$ (we denote by $\mathbb{R}$ the set of all nonnegative real numbers). A *path* in $\mathcal{T}$ is an infinite sequence of pairs $\langle w_0, \delta_0 \rangle, \langle w_1, \delta_1 \rangle, \ldots$, such that for all $i \geq 0$, we have $\langle w_i, \delta_i, w_{i+1} \rangle \in R$. A *timed word* is an infinite sequence $\tau \in (2^{AP} \times \mathbb{R})^\omega$. We sometimes refer to a timed word as a function $\tau : \mathbb{N} \to 2^{AP} \times \mathbb{R}$ and use $\tau_1(i)$ and $\tau_2(i)$ to refer to the i'th event and duration, respectively, in τ.

2.2 Timed Automata

We represent real-time programs by timed automata. We now define timed automata and the timed structures induced by them.

Given a set C of clocks, a *clock environment* $\mathcal{E} : C \to \mathbb{R}$ assigns to each clock a nonnegative real value. Given a clock environment $\mathcal{E}$, a set $S \subseteq C$ of clocks and a nonnegative real $\delta \in \mathbb{R}$, we define $progress(\mathcal{E}, S, \delta)$ as the clock environment $\mathcal{E}'$ that resets all clocks in S and advances all clocks in $S \setminus C$ by δ; that is, for all $c \in C$, we have

$$\mathcal{E}'(c) = \begin{cases} 0 & \text{if } c \in S, \\ \mathcal{E}(c) + \delta & \text{if } c \notin S. \end{cases}$$

For two clock environments $\mathcal{E}$ and $\mathcal{E}'$, we say that $\mathcal{E} < \mathcal{E}'$ iff for every clock $c \in C$, we have $\mathcal{E}(c) < \mathcal{E}'(c)$. We use $\mathcal{E}^0$ to denote the clock environment that assigns 0 to all clocks. For a set C of clocks, a formula (also referred sometimes as *clock constraint*) in $guard(C)$ is one of the following:

- **true**, **false**, or $c \sim v$, where $c \in C$, $v \in \mathbb{N}$, and $\sim \in \{\geq, >, \leq, <\}$,
- $\theta_1 \vee \theta_2$ or $\theta_1 \wedge \theta_2$, where θ_1 and θ_2 are formulas in $guard(C)$.

A *timed automaton* is a tuple $\mathcal{U} = \langle AP, C, L, E, P, inv, l^0 \rangle$, where

- AP is a finite set of atomic propositions,
- C is a finite set of program clocks,
- L is a finite set of locations,
- $E : L \to 2^{guard(C) \times 2^C \times L}$ is a nondeterministic transition function,
- $P : L \to 2^{AP}$ assigns to each location a set of atomic propositions,
- $inv : L \to guard(C)$ assigns to each location an invariant,
- $l^0 \in L$ is an initial location.

A *position* of $\mathcal{U}$ is a pair $\langle l, \mathcal{E} \rangle \in L \times \mathbb{R}^C$; that is, a position describes a location and a clock environment. Given a position $\langle l, \mathcal{E} \rangle$ and a time delay $\delta \in \mathbb{R}$, we say that the position $\langle l', \mathcal{E}' \rangle$ is a δ-*successor* of $\langle l, \mathcal{E} \rangle$ iff there exists a triple $\langle \theta, S, l' \rangle \in E(l)$ such that the following three conditions hold:

1. $progress(\mathcal{E}, \emptyset, \delta) \models \theta$.
2. For every nonnegative real $\delta' < \delta$, we have $progress(\mathcal{E}, \emptyset, \delta') \models inv(l)$.
3. $\mathcal{E}' = progress(\mathcal{E}, S, \delta)$.

Timed automata run on timed words. A *run* r of a timed automaton on the timed word $\mathcal{U}$ is an infinite sequence of positions of $\mathcal{U}$. Thus, $r \in (L \times \mathbb{R}^C)^\omega$. We sometimes refer to a run as a function $r : \mathbb{N} \to L \times \mathbb{R}^C$. Given a timed word $\tau : \mathbb{N} \to 2^{AP} \times \mathbb{R}$, a run r of $\mathcal{U}$ on τ satisfies the following. For every $i \geq 0$, let $r(i) = \langle l_i, \mathcal{E}_i \rangle$. Then:

- $l_0 = l^0$ and $\mathcal{E}_0 = \mathcal{E}^0$.
- For every $i \geq 0$, we have $P(l_i) = \tau_1(i)$.
- For every $i \geq 0$, we have $\langle l_{i+1}, \mathcal{E}_{i+1} \rangle$ is a $\tau_2(i)$-successor of $\langle l_i, \mathcal{E}_i \rangle$.

We say that $\mathcal{U}$ *accepts* the timed word τ iff there exists a run of $\mathcal{U}$ on τ. The *language* of $\mathcal{U}$ is the set of all timed words that are accepted by $\mathcal{U}$. Each timed automaton induces a timed structure. Formally, the *timed structure of* $\mathcal{U}$ is

$$\mathcal{T}(\mathcal{U}) = \langle AP, L \times \mathbb{R}^C, R, \{\langle l^0, \mathcal{E}^0 \rangle\}, \sigma \rangle,$$

where R and σ are defined as follows:

- $R(\langle l, \mathcal{E}\rangle, \delta, \langle l', \mathcal{E}'\rangle)$ iff $\langle l', \mathcal{E}'\rangle$ is a δ-successor of $\langle l, \mathcal{E}\rangle$.
- For all states $\langle l, \mathcal{E}\rangle \in L \times \mathbb{R}^C$, we have $\sigma(\langle l, \mathcal{E}\rangle) = P(l)$.

Note that the state set of $\mathcal{T}(\mathcal{U})$ is infinite and that $\mathcal{T}(\mathcal{U})$ may have an infinite branching degree. It is easy to see that a timed word is accepted by $\mathcal{U}$ iff it is induced by a path, starting at $\langle l^0, \mathcal{E}^0\rangle$, in $\mathcal{T}(\mathcal{U})$.

2.3 The Real-time Branching Temporal Logic TCTL

We specify properties of real-time programs using real-time temporal logics. We consider here TCTL, a real-time extension of the branching temporal logic CTL with clocks [ACD93]. Formulas of TCTL are defined with respect to the sets AP and $C_{\mathcal{U}}$ of the program's atomic propositions and clocks, respectively, and a set C_ψ of specification clocks. We consider TCTL formulas in positive normal form in which negation may apply to atomic propositions only. Given $AP, C_{\mathcal{U}}$, and C_ψ, a formula of TCTL is one of the following:

- **true**, **false**, p, or $\neg p$, where $p \in AP$.
- $c \sim v$, where $c \in C_{\mathcal{U}} \cup C_\psi$, $v \in \mathbb{N}$, and $\sim \in \{\geq, >, \leq, <\}$.
- $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $E\varphi_1 U \varphi_2$, $A\varphi_1 U \varphi_2$, $E\varphi_1 \tilde{U} \varphi_2$, or $A\varphi_1 \tilde{U} \varphi_2$, where φ_1 and φ_2 are TCTL formulas.
- $c.\varphi$, where $c \in C_\psi$ and φ is a TCTL formula.

The temporal operator $\tilde{U}$ is dual to the U ("until") operator. For example, the formula $E\varphi_1 \tilde{U} \varphi_2$ is equivalent to the formula $\neg A(\neg\varphi_1) U (\neg\varphi_2)$. In addition, we use the usual $\rightarrow$ ("implies"), F ("eventually"), and G ("always") abbreviations. The reset quantifier $c.\varphi$ *binds* all free occurrences of c in φ. We denote by $free(\psi) \subseteq C_{\mathcal{U}} \cup C_\psi$ the set of clocks free in ψ (i.e., these that have a non-bound occurrence). We denote by $bound(\psi) \subseteq C_\psi$ the set of all clocks bound in ψ (i.e., clocks c for which $c.\varphi$ is a subformula of ψ). For example, the formula $\psi = (c_1 = 5) \vee c_2.AF(c_2 > 0)$ has $free(\psi) = \{c_1\}$ and $bound(\psi) = \{c_2\}$. By renaming clocks we can make sure that no occurrence of a clock in ψ is bound to more than one reset quantifier and that $bound(\psi)$ and $free(\psi)$ are disjoint. A TCTL formula ψ is *closed* iff $free(\psi) = \emptyset$. We denote by $cl(\psi)$ the set of all subformulas of ψ. It is easy to see that the size of $cl(\psi)$ is bounded by the length of ψ.

TCTL formulas are interpreted over pairs that consist of a state of a timed structure $\mathcal{T}(\mathcal{U})$, where $C_{\mathcal{U}}$ is the set of $\mathcal{U}$'s clocks, and a clock environment $\mathcal{E} : C_{\mathcal{U}} \cup C_\psi \rightarrow \mathbb{R}$. We use $w, \mathcal{E} \models \varphi$ to indicate that a formula φ holds at state w with clock environment $\mathcal{E}$ (with respect to the given timed structure $\mathcal{T}$). A formal definition of the relation $\models$ can be found in [HNSY94]. We will define later the semantics of TCTL formulas when interpreted over quotient graphs induced by timed structures.

For a timed structure $\mathcal{T}$ and a TCTL formula ψ, we say that $\mathcal{T} \models \psi$ iff $\langle w^0, \mathcal{E}^0\rangle \models \psi$. The *model-checking problem* for TCTL is defined as follows: given a timed automaton $\mathcal{U}$ and a closed TCTL formula ψ, determine whether $\mathcal{T}(\mathcal{U}) \models \psi$ (denoted $\mathcal{U} \models \psi$).

2.4 Regions and Region Structures

Each set C of clocks induces infinitely many clock environments. Given a set C of clocks and a set G of clock constraints in $guard(C)$, we can partition the clock environments in $\mathcal{R}^C$ into finitely many equivalence classes such that all clock environments in the same class

are indistinguishable by clock constraints in G. It was proven in [AD94] that a sufficient condition for two environment clocks to be indistinguishable is agreement on the integral parts of all clocks values and agreement on the ordering of the fractional parts of all clock values. This leads to the following definition of *regions*. For $x \in \mathbb{R}$, let $\lfloor x \rfloor$ and $\langle x \rangle$ denote the integer and the fractional parts of x, respectively. Also, for each clock $c \in C$, let v_c be the largest integer v for which $x \sim v$ is a subformula of some clock constraint in G. We define an equivalent relation $\approx \subseteq \mathbb{R}^C \times \mathbb{R}^C$. For two clock environments $\mathcal{E}$ and $\mathcal{E}'$, we have $\mathcal{E} \approx \mathcal{E}'$ iff the following three conditions hold:

1. For all $c \in C$, either $\lfloor \mathcal{E}(c) \rfloor = \lfloor \mathcal{E}'(c) \rfloor$, or $\mathcal{E}(c) > v_c$ and $\mathcal{E}'(c) > v_c$.
2. For all $\{c, d\} \subseteq C$ with $\mathcal{E}(c) \leq v_c$ and $\mathcal{E}(d) \leq v_d$, we have $\langle \mathcal{E}(c) \rangle \leq \langle \mathcal{E}(d) \rangle$ iff $\langle \mathcal{E}'(c) \rangle \leq \langle \mathcal{E}'(d) \rangle$.
3. For all $c \in C$ with $\mathcal{E}(c) \leq v_c$, we have $\langle \mathcal{E}(c) \rangle = 0$ iff $\langle \mathcal{E}'(c) \rangle = 0$.

We now define a region to be an equivalence class of the relation $\approx$.

We denote the set of all regions induced by C and G by $\Upsilon(C, G)$. When C and G are clear or not important, we use only Υ. Let $rep : \Upsilon \to \mathbb{R}^C$ map each region to an arbitrary representative clock environment in it. We represent a region π by $rep(\pi)$. A clock environments $\mathcal{E}$ then belongs to π iff $\mathcal{E} \approx rep(\pi)$. We sometime represent a region also by a finite set of clock constraints (e.g., $[x = 1; 2 < z < 3]$). A clock environment $\mathcal{E}$ then belongs to π iff it satisfies all its clock constraints. Following the definition of regions, the constraints that represent π specify the integral part of all clocks, the order among the fractional parts, and whether they are equal to 0.

Lemma 1. [AD94] *The number of regions in $\Upsilon(C, G)$ is bounded by $|C|! \cdot 2^{|C|} \cdot \prod_{c \in C}(2v_c + 2)$.*

For a region π and a formula $\varphi \in G$, we say that π satisfies φ (denoted $\pi \models \varphi$) iff $rep(\pi)$ satisfies φ. Note that by the definition of regions, π satisfies φ iff all clock environments in π satisfy φ. Given a clock constraint $c \sim v$ in G, let $reg(c \sim v)$ be the union of all regions that satisfy $c \sim v$.

Each region has a unique successor region. Intuitively, the successor of a region π is obtained from π by letting time pass. The function $succ : \Upsilon \to \Upsilon$ maps a region π to its successor region. For a region π and a clock environment $\mathcal{E}$, we have that $\mathcal{E} \in succ(\pi)$ iff $rep(\pi) \not\approx \mathcal{E}$, $rep(\pi) < \mathcal{E}$, and for every clock environment $\mathcal{E}'$ with $rep(\pi) < \mathcal{E}' < \mathcal{E}$, we have $\mathcal{E}' \approx rep(\pi)$ or $\mathcal{E}' \approx \mathcal{E}$. So, for example, if π has a clock constraint $c = v$, for $v \neq v_c$, its successor has a clock constraint $v < c < v + 1$. If $v = v_c$, then the successor has a clock constraint $c > v$, reflecting the fact that once c is larger than v_c, we are no longer interested in how large it is.

For $\pi \in \Upsilon(C, G)$ and a set S of clocks, we denote by $\pi[S := 0]$ the region obtained from π by resetting all clocks in S. That is, $\pi[S := 0]$ contains the clock environment $progress(rep(\pi), S, 0)$. For a guard $\varphi \in G$ and $S \subseteq C$ we denote by $\varphi[S := 0]$ the guard obtained from φ by replacing with 0 all clocks in S. Also, let $\Upsilon[0]$ denote the region where all clocks in C are set to 0.

Consider a timed automaton $\mathcal{U} = \langle AP, C_{\mathcal{U}}, L, E, P, inv, l^0 \rangle$ and a TCTL formula ψ over $AP, C_{\mathcal{U}}$, and a set C_ψ of specification clocks. Let G be the set of clock constraints in $\mathcal{U}$ and ψ, and let $\Upsilon = \Upsilon(C_{\mathcal{U}} \cup C_\psi, G)$ be the set of regions induced by $\mathcal{U}$ and ψ. We define a *region position* of $\mathcal{U}$ as a pair $\langle l, \pi \rangle \in L \times \Upsilon$. When we say that $\mathcal{U}$ is in region position $\langle l, \pi \rangle$, we mean that $\mathcal{U}$ is in location l and that its clock environment is in π. We say that a region position $\langle l, \pi \rangle$ is *admissible* iff π satisfies $inv(l)$. We know that the automaton $\mathcal{U}$

can be only in admissible region positions. Moreover, when $\mathcal{U}$ is in region position $\langle l, \pi \rangle$, we know what its possible next region positions are: the automaton $\mathcal{U}$ can either take an *edge transition* and move to another location, possibly resetting some clocks, or take a *time transition* and stay in l while the values of the clocks change and meet the successor region. This leads to the following definition of the *region structure* $\mathcal{R}(\mathcal{U}, \psi)$ induced by $\mathcal{U}$ and ψ.

We define $\mathcal{R}(\mathcal{U}, \psi) = \langle AP \cup \Upsilon, W, R, w^0, \sigma \rangle$ to be the following Kripke structure:

- The set $W \subseteq L \times \Upsilon$ of states consists of all the admissible positions of $\mathcal{U}$.
- $R(\langle l, \pi \rangle, \langle l', \pi' \rangle)$ iff one of the following two conditions holds:
  1. $l = l'$ and $\pi' = succ(\pi)$. These transitions correspond to time transitions in $\mathcal{U}$ and we call them *time transitions*.
  2. There exists a transition $\langle \theta, S, l' \rangle \in E(l)$ such that π satisfies θ and $\pi' = \pi[S := 0]$. These transitions correspond to edge transitions in $\mathcal{U}$ and we call them *edge transitions*.
- The initial state w^0 is $\langle l^0, \Upsilon[0] \rangle$. If w^0 is not admissible, the automaton $\mathcal{U}$ is not an interesting real-time program.
- For all states $\langle l, \pi \rangle \in W$, we have $\sigma(\langle l, \pi \rangle) = P(l) \cup \{\pi\}$.

We can interpret a TCTL formula ψ with respect to a state in the timed automaton $\mathcal{U}$ and a clock environment for the specification clocks by means of the region structure $\mathcal{R}(\mathcal{U}, \psi)$. We use $\langle l, \pi \rangle \models \varphi$ to indicate that a subformula φ of ψ holds at state $\langle l, \pi \rangle$ of $\mathcal{R}(\mathcal{U}, \psi)$. Note that as π refers to the values of the clocks in both $C_{\mathcal{U}}$ and C_ψ, we do not need a clock environment for the specification clocks. The relation $\models$ is defined inductively as follows:

- For all l and π, we have $\langle l, \pi \rangle \models \mathbf{true}$ and $\langle l, \pi \rangle \not\models \mathbf{false}$.
- For $p \in AP$, we have $\langle l, \pi \rangle \models p$ iff $p \in \sigma(\langle l, \pi \rangle)$, and $\langle l, \pi \rangle \models \neg p$ iff $p \notin \sigma(\langle l, \pi \rangle)$.
- For $c \in C$, $v \in \mathbb{N}$, and $\sim \in \{\geq, >, \leq, <\}$, we have $\langle l, \pi \rangle \models c \sim v$ iff $\pi \models c \sim v$.
- $\langle l, \pi \rangle \models \varphi_1 \vee \varphi_2$ iff $\langle l, \pi \rangle \models \varphi_1$ or $\langle l, \pi \rangle \models \varphi_2$.
- $\langle l, \pi \rangle \models \varphi_1 \wedge \varphi_2$ iff $\langle l, \pi \rangle \models \varphi_1$ and $\langle l, \pi \rangle \models \varphi_2$.
- $\langle l, \pi \rangle \models E\varphi_1 U \varphi_2$ iff there exists a path $\langle l_0, \pi_0 \rangle, \langle l_1, \pi_1 \rangle \ldots$ in $\mathcal{R}(\mathcal{U}, \psi)$ with $\langle l_0, \pi_0 \rangle = \langle l, \pi \rangle$, and there exists $i \geq 0$ such that $\langle l_i, \pi_i \rangle \models \varphi_2$, and for all $0 \leq j < i$ we have $\langle l_j, \pi_j \rangle \models \varphi_1$.
- $\langle l, \pi \rangle \models A\varphi_1 U \varphi_2$ iff for every path $\langle l_0, \pi_0 \rangle, \langle l_1, \pi_1 \rangle \ldots$ in $\mathcal{R}(\mathcal{U}, \psi)$ with $\langle l_0, \pi_0 \rangle = \langle l, \pi \rangle$, there exists $i \geq 0$ such that $\langle l_i, \pi_i \rangle \models \varphi_2$ and for all $0 \leq j < i$ we have $\langle l_j, \pi_j \rangle \models \varphi_1$.
- $\langle l, \pi \rangle \models E\varphi_1 \tilde{U} \varphi_2$ iff there exists a path $\langle l_0, \pi_0 \rangle, \langle l_1, \pi_1 \rangle \ldots$ in $\mathcal{R}(\mathcal{U}, \psi)$ with $\langle l_0, \pi_0 \rangle = \langle l, \pi \rangle$ such that for every $i \geq 0$ for which $\langle l_i, \pi_i \rangle \not\models \varphi_2$, there exists $j < i$ such that $\langle l_j, \pi_j \rangle \models \varphi_1$.
- $\langle l, \pi \rangle \models A\varphi_1 \tilde{U} \varphi_2$ iff for every path $\langle l_1, \pi_1 \rangle, \langle l_2, \pi_2 \rangle \ldots$ in $\mathcal{R}(\mathcal{U}, \psi)$ with $\langle l_1, \pi_1 \rangle = \langle l, \pi \rangle$ and for every $i \geq 0$ for which $\langle l_i, \pi_i \rangle \not\models \varphi_2$, there exists $j < i$ such that $\langle l_j, \pi_j \rangle \models \varphi_1$.
- $\langle l, \pi \rangle \models c.\varphi$ iff $\langle l, \pi[c := 0] \rangle \models \varphi$.

We say that $\mathcal{R}(\mathcal{U}, \psi) \models \psi$ iff $w^0 \models \psi$. Several works on real-time temporal logics consider a more elaborated semantic for TCTL, where path quantification ranges only over paths for which time diverges [HNSY94]. As we discuss in Section 4, our algorithm can be easily extended to handle this semantics as well.

By the definition of regions, we have the following.

Theorem 2. [HNSY94] *For every timed automaton $\mathcal{U}$ and TCTL formula ψ, we have $\mathcal{U} \models \psi$ iff $\mathcal{R}(\mathcal{U}, \psi) \models \psi$.*

3 Reducing TCTL Model Checking to CTL Model Checking

3.1 Untiming the Program

Consider a timed automaton $\mathcal{U} = \langle AP, C_{\mathcal{U}}, L, E, P, inv, l^0 \rangle$ and a TCTL formula ψ over $AP, C_{\mathcal{U}}$, and a set C_ψ of specification clocks. Let $\mathcal{R}(\mathcal{U}, \psi)$ be the region structure induced by $\mathcal{U}$ and ψ. Many states in $\mathcal{R}(\mathcal{U}, \psi)$ are not reachable. Indeed, the transitions in $\mathcal{R}(\mathcal{U}, \psi)$ may reset some of the clocks in $C_{\mathcal{U}}$ but can never reset clocks in C_ψ. We need these unreachable states because ψ may contain reset quantifiers that cause the course of the evaluation of ψ to "jump" into these states. When we untime the program, we make these states reachable. Below we define the *Kripke structure* $\mathcal{K}(\mathcal{U}, \psi)$ induced by $\mathcal{U}$ and ψ. The structure $\mathcal{K}(\mathcal{U}, \psi)$ is very similar to $\mathcal{R}(\mathcal{U}, \psi)$. Each state w in $\mathcal{K}(\mathcal{U}, \psi)$ corresponds to two states $\langle w, \mathbf{T} \rangle$ and $\langle w, \mathbf{E} \rangle$ in $\mathcal{R}(\mathcal{U}, \psi)$. The copy of w annotated with $\mathbf{T}$ can be reached only by time transitions. The copy of w annotated with $\mathbf{E}$ can be reached only by edge transitions. Time transitions in $\mathcal{K}(\mathcal{U}, \psi)$ may reset an arbitrary subset of the clocks in C_ψ. Formally,

$$\mathcal{K}(\mathcal{U}, \psi) = \langle AP \cup \Upsilon \cup \{\mathbf{T}\}, W \times \{\mathbf{T}, \mathbf{E}\}, R, \langle w^0, \mathbf{E} \rangle, \varsigma \rangle,$$

where W, w^0, and are as in $\mathcal{R}(\mathcal{U}, \psi)$, and R and ς are defined as follows:

- $R(\langle l, \pi, b \rangle, \langle l', \pi', b' \rangle)$ iff one of the following two conditions holds:
  1. $b' = \mathbf{T}$, $l = l'$, and there exists a set $S_\psi \subseteq C_\psi$ of specification clocks such that $\pi' = succ(\pi)[S_\psi := 0]$. These transitions correspond to time transitions in $\mathcal{U}$ and we call them *time transitions*.
  2. $b' = \mathbf{E}$ and there exists a transition $\langle \theta, S, l' \rangle \in E(l)$ such that π satisfies θ and $\pi' = \pi[S := 0]$. These transitions correspond to edge transitions in $\mathcal{U}$ and we call them *edge transitions*.
- The set of atomic propositions that hold in each state is as the one in $\mathcal{R}(\mathcal{U}, \psi)$, only that we add the atomic proposition $\mathbf{T}$ to states in $W \times \{\mathbf{T}\}$. Thus, $\varsigma(\langle w, b \rangle)$ is $\sigma(w)$ if $b = \mathbf{E}$, and is $\sigma(w) \cup \{\mathbf{T}\}$ if $b = \mathbf{T}$.

Note that the specification clocks are changed only in time transitions, when each specification clock is either advanced by the same amount as the program clocks, or it is reset. The duplication of the states and the new atomic proposition $\mathbf{T}$ enable us to distinguish between states that are reached by a time transition and states that are reached by an edge transition. Following Lemma 1, the size of $\mathcal{K}(\mathcal{U}, \psi)$ is exponential in the size of $\mathcal{U}$ and the length of ψ.

3.2 Untiming the Specification

We define a function

$$untime: \text{ TCTL formulas } \rightarrow \text{ CTL formulas}$$

such that for every timed automaton $\mathcal{U}$ and TCTL formula ψ, we have $\mathcal{U} \models \psi$ iff $\mathcal{K}(\mathcal{U}, \psi) \models untime(\psi)$. We define the function $untime$ by means of a function

$$f: \text{ TCTL formulas } \times \text{ sets of specification clocks } \rightarrow \text{ CTL formulas.}$$

For a TCTL formula ψ, the CTL formula $untime(\psi)$ is then the formula $f(\psi, \emptyset)$.

For a set $S \subseteq C$ of clocks, we use $S = 0$ as an abbreviation for $\bigwedge_{c \in S}(c = 0)$ and use $S > 0$ as an abbreviation for $\bigwedge_{c \in S}(c > 0)$. Note that when $S = \emptyset$, the formulas $S = 0$ and $S > 0$ evaluate to **true**. Consider a path ρ in $\mathcal{K}(\mathcal{U}, \psi)$. For sets S_1 and S_2 of clocks, we use $fair(S_1, S_2)$ to abbreviate a path formula stating that the clocks in S_1 are never reset and the clocks in S_2 are always reset along the path ρ. That is,

$$fair(S_1, S_2) = XG(\mathbf{T} \rightarrow (S_1 > 0)) \wedge G(S_2 = 0).$$

Given a TCTL formula φ and a set $S \subseteq C_\varphi$ of clocks, we define $f(\varphi, S)$ by induction on the structure of φ as follows (we first present the mapping into CTL* formulas and then translate them, in Section 3.4, to CTL).

- $f(\mathbf{true}, S) = \mathbf{true}$ and $f(\mathbf{false}, S) = \mathbf{false}$.
- For $p \in AP$, we have $f(p, S) = p$ and $f(\neg p, S) = \neg p$.
- For a clock constraint $c \sim v$, we have $f(c \sim v, S) = reg((c \sim v)[S := 0])$.
- $f(\varphi_1 \vee \varphi_2, S) = f(\varphi_1, S) \vee f(\varphi_2, S)$.
- $f(\varphi_1 \wedge \varphi_2, S) = f(\varphi_1, S) \wedge f(\varphi_2, S)$.
- $f(E\varphi_1 U \varphi_2, S) = E[fair(S, bound(E\varphi_1 U \varphi_2)) \wedge f(\varphi_1, \emptyset) U f(\varphi_2, \emptyset)]$.
- $f(A\varphi_1 U \varphi_2, S) = A[fair(S, bound(A\varphi_1 U \varphi_2)) \rightarrow f(\varphi_1, \emptyset) U f(\varphi_2, \emptyset)]$.
- $f(E\varphi_1 \tilde{U} \varphi_2, S) = E[fair(S, bound(E\varphi_1 \tilde{U} \varphi_2)) \wedge f(\varphi_1, \emptyset) \tilde{U} f(\varphi_2, \emptyset)]$.
- $f(A\varphi_1 \tilde{U} \varphi_2, S) = A[fair(S, bound(A\varphi_1 \tilde{U} \varphi_2)) \rightarrow f(\varphi_1, \emptyset) \tilde{U} f(\varphi_2, \emptyset)]$.
- $f(c.\varphi_1, S) = f(\varphi_1, S \cup \{c\})$.

Intuitively, the set S in $f(\varphi, S)$ contains all the free clocks in φ that should be reset and then never reset again once we come to evaluate φ. When we evaluate a formula φ, path quantification ranges only over paths in which the clocks in $bind(\varphi)$ are always reset. This restricted quantification is imposed by the second conjunct in the path formula *fair*. Consider a clock $c \in S$. We know that c enters S as a result of being a binding clock in a formula of the form $c.\varphi_1$. Hence, when c enters S is is not free and therefore, by the above rules, it is reset. We "release" the clock c and path quantification becomes restricted to paths in which c and the other clocks in S are never reset (this is imposed by the first conjunct in *fair*) and in which the clocks in $bind(\varphi_1)$ remain always reset.

If φ is a clock constraint, then releasing c is done by simply assigning 0 to c. If φ is of the form $E\varphi_1 U \varphi_2$, $A\varphi_1 U \varphi_2$, $E\varphi_1 \tilde{U} \varphi_2$, or $A\varphi_1 \tilde{U} \varphi_2$, then releasing c is done by updating the parameters of the formula *fair*. For example, the formula

$$\psi = AG[c.req \rightarrow AF(grant \wedge (c \leq 2))],$$

mentioned earlier in the introduction, has

$$untime(\psi) = A[Gc_0 \rightarrow G(req \rightarrow A[XG(T \rightarrow \neg c_0) \rightarrow F(grant \wedge c_{\leq 2})],$$

where c_0 and $c_{\leq 2}$ abbreviate $reg(c = 0)$ and $reg(c \leq 2)$, respectively.

It is easy to see that the length of $untime(\psi)$ is linear in the length of ψ (we ignore quadratic blow-up caused by specifying sets of clocks and sets of regions in *fair*; such a blow-up can be easily handled by new atomic propositions).

3.3 Correctness of the Reduction

We now prove the correctness of the reduction. Along the proof, we use $\langle l, \pi \rangle \models_{\mathcal{R}} \psi$ to indicate that the state $\langle l, \pi \rangle$ satisfies the TCTL formula ψ in the region structure $\mathcal{R}(\mathcal{U}, \psi)$ and we use $\langle l, \pi \rangle \models_{\mathcal{K}} \varphi$ to indicate that the states $\langle l, \pi, \mathbf{E} \rangle$ and $\langle l, \pi, \mathbf{T} \rangle$ satisfy the CTL formula φ in the Kripke structure $\mathcal{K}(\mathcal{U}, \psi)$. Note that as $\langle l, \pi, \mathbf{E} \rangle$ and $\langle l, \pi, \mathbf{T} \rangle$ have exactly the same future, and they differ only in the value of $\mathbf{T}$, they agree on satisfaction of CTL formulas that do not refer to the value of $\mathbf{T}$ in the present.

Theorem 3. *For every timed automaton $\mathcal{U}$, TCTL formula ψ, location l of $\mathcal{U}$, subformula φ of ψ, region $\pi \in \Upsilon$, and set $S \subseteq C_\psi$ of specification clocks such that $S \cap bound(\varphi) = \emptyset$, the following are equivalent:*

1. $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi$.
2. $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi, S)$.

Proof. The proof proceeds by induction on the structure of φ.

- The cases where φ is of the form **true**, **false**, p, or $\neg p$ are immediate.

- For φ that is a clock constraint $c \sim v$, we have $f(c \sim v, S) = reg((c \sim v)[S := 0])$. Since the clocks in S are reset anyway, then $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} (c \sim v)$ holds iff $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} (c \sim v)[S := 0]$. Since the satisfaction of $c \sim v$ depends only on the value of c and is independent of l, then the latter, according to the definition of the mapping reg, holds iff $\langle l, \pi[S := 0] \rangle \models_{\mathcal{K}} reg((c \sim v)[S := 0])$; thus iff $\langle l, \pi[S := 0] \rangle \models_{\mathcal{K}} f(c \sim v, S)$. Finally, as $bound(\varphi) = \emptyset$, the latter holds iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(c \sim v, S)$, and we are done.

- For $\varphi = \varphi_1 \wedge \varphi_2$, we have that $f(\varphi, S) = f(\varphi_1, S) \wedge f(\varphi_2, S)$. By the semantics of TCTL, $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi_1 \wedge \varphi_2$ iff $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi_1$ and $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi_2$. Since $S \cap bound(\varphi) = \emptyset$, then $S \cap bound(\varphi_1) = \emptyset$. Therefore, the induction hypothesis is applicable, and $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi_1$ iff $\langle l, \pi[S \cup bound(\varphi_1) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S)$.

Consider the set of clocks $bound(\varphi) \setminus bound(\varphi_1)$. By the syntax of TCTL, these clocks do not appear in φ_1, and therefore, as $S \cap bind(\varphi) = \emptyset$, they do not appear in in $f(\varphi_1, S)$ either. Hence $\langle l, \pi[S \cup bound(\varphi_1) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S)$ iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S)$. Similarly, $\langle l, \pi[S \cup bound(\varphi_2) := 0] \rangle \models_{\mathcal{K}} f(\varphi_2, S)$ iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_2, S)$. We thus have that $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi_1 \wedge \varphi_2$ iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S)$ and $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_2, S)$. This holds iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S) \wedge f(\varphi_2, S)$. This, by the definition of f, holds iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1 \wedge \varphi_2, S)$, and we are done.

The proof is similar for φ of the form $\varphi_1 \vee \varphi_2$.

- For $\varphi = E\varphi_1 U \varphi_2$, we have $f(E\varphi_1 U \varphi_2, S) = E[fair(S, bound(\varphi)) \wedge f(\varphi_1, \emptyset) U f(\varphi_2, \emptyset)]$. Consider a state $\langle l, \pi \rangle$ in $\mathcal{R}(\mathcal{U}, \psi)$. Assume first that $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi$. Then, by the semantic of TCTL, there is a path $\langle l_0, \pi_0 \rangle, \langle l_1, \pi_1 \rangle \ldots$ in $\mathcal{R}(\mathcal{U}, \psi)$ with $\langle l_0, \pi_0 \rangle = \langle l, \pi[S := 0] \rangle$, and there exists $i \geq 0$ such that $\langle l_i, \pi_i \rangle \models_{\mathcal{R}} \varphi_2$, and for all $0 \leq j < i$ we have $\langle l_j, \pi_j \rangle \models_{\mathcal{R}} \varphi_1$. By the induction hypothesis (applied with $\varphi = \varphi_2$, $l = l_i, \pi = \pi_i$, and $S = \emptyset$), $\langle l_i, \pi_i \rangle \models_{\mathcal{R}} \varphi_2$ iff $\langle l_i, \pi_i[bound(\varphi_2) := 0] \rangle \models_{\mathcal{K}} f(\varphi_2, \emptyset)$. Similarly, for all $0 \leq j < i$, we have that $\langle l_j, \pi_j \rangle \models_{\mathcal{R}} \varphi_1$ iff $\langle l_j, \pi_j[bound(\varphi_1) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, \emptyset)$. As in the $\varphi_1 \wedge \varphi_2$ case, this implies, that $\langle l_i, \pi_i \rangle \models_{\mathcal{R}} \varphi_2$ iff $\langle l_i, \pi_i[bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_2, \emptyset)$, and for all $0 \leq j < i$, we have that $\langle l_j, \pi_j \rangle \models \varphi_1$ iff $\langle l_j, \pi_j[bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, \emptyset)$. Hence, the induction hypothesis is applicable as follows. Consider the sequence of regions $\eta = \eta_0, \eta_1, \ldots$ and the sequence of attributions $b_0, b_1, \ldots \in \{\mathbf{T}, \mathbf{E}\}^\omega$, where

- $\eta_0 = \pi[S \cup bound(\varphi) := 0]$ and $b_0 = \mathbf{E}$.
- For every $i \geq 0$, the region η_{i+1} and the attribution b_{i+1} are defined as follows.
  - If the transition in $\mathcal{R}(\mathcal{U}, \psi)$ from $\langle l_i, \pi_i \rangle$ to $\langle l_{i+1}, \pi_{i+1} \rangle$ is an edge transition, in which case there exists $S_{\mathcal{U}} \subseteq C_{\mathcal{U}}$ for which $\pi_{i+1} = \pi_i[S_{\mathcal{U}} := 0]$, then $\eta_{i+1} = \eta_i[S_{\mathcal{U}} := 0]$ and $b_{i+1} = \mathbf{E}$.
  - If the transition in $\mathcal{R}(\mathcal{U}, \psi)$ from $\langle l_i, \pi_i \rangle$ to $\langle l_{i+1}, \pi_{i+1} \rangle$ is a time transition, then $\eta_{i+1} = succ(\eta_i)[bound(\varphi) := 0]$ and $B_{i+1} = \mathbf{T}$.

Note that for every $k \geq 0$, we have $\eta_k = \pi_k[bound(\varphi) := 0]$. Thus, clearly, for all $k \geq 0$, we have that $\eta_k \models (bound(\varphi) = 0)$. Also, for all $k \geq 1$, since the only specification clocks that are reset along ρ are these in $bind(\varphi)$ and since $S \cap bind(\varphi) = \emptyset$, we have that $\eta_k \models \mathbf{T} \rightarrow (S > 0)$. Indeed, the value of a clock that is not reset in a time transition must become greater than 0. Since $bound(\varphi) \subseteq C_\psi$ then, by the definition of $\mathcal{K}(\mathcal{U}, \psi)$, the sequence $\rho = \langle l_0, \eta_0, b_0 \rangle, \langle l_1, \eta_1, b_1 \rangle, \ldots$ is a path in $\mathcal{K}(\mathcal{U}, \psi)$. As detailed above, by the induction hypothesis, we have that $\langle l_i, \eta_i \rangle \models_{\mathcal{K}} f(\varphi_2, \emptyset)$, and for all $0 \leq j < i$, we have that $\langle l_j, \eta_j \rangle \models f(\varphi_1, \emptyset)$. In addition, ρ satisfies $fair(S, bound(\varphi))$. Hence, $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} E[fair(S, bound(\varphi)) \wedge f(\varphi_1, \emptyset) U f(\varphi_2, \emptyset)]$, and we are done.

Assume now that $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi, S)$. Therefore, there exists a path $\langle l_0, \pi_0, b_0 \rangle, \langle l_1, \pi_1, b_1 \rangle, \ldots$ in $\mathcal{K}(\mathcal{U}, \psi)$ with $\langle l_0, \pi_0, b_0 \rangle = \langle l, \pi[bound(\varphi) \cup S := 0], \mathbf{E} \rangle$, such that the following hold. First, for all $k \geq 1$, we have that $\pi_k \models \mathbf{T} \rightarrow (S > 0)$. Second, for all $k \geq 0$, we have that $\pi_k \models (bound(\varphi) = 0)$. In addition, there exists $i \geq 0$ such that $\langle l_i, \pi_i, b_i \rangle \models f(\varphi_2, \emptyset)$, and for all $0 \leq j < i$, we have $\langle l_j, \pi_j, b_j \rangle \models f(\varphi_1, \emptyset)$. Consider the sequence of regions $\eta = \eta_0, \eta_1, \ldots$ defined as follows.

- $\eta_0 = \pi[S := 0]$.
- For every $i \geq 0$, the region η_{i+1} is defined as follows.
  - If $b_{i+1} = \mathbf{E}$, in which case there exists $S_{\mathcal{U}} \subseteq C_{\mathcal{U}}$ for which $\pi_{i+1} = \pi_i[S_{\mathcal{U}} := 0]$, then $\eta_{i+1} = \eta_i[S_{\mathcal{U}} := 0]$.
  - If $b_{i+1} = \mathbf{T}$, then $\eta_{i+1} = succ(\eta_i)$.

Note that for every $k \geq 0$, we have $\pi_k = \eta_k[bound(\varphi) := 0]$. Also, since no specification clocks are reset along η, it is guaranteed that the sequence $\langle l_0, \eta_0 \rangle, \langle l_1, \eta_1 \rangle, \ldots$ is a path in $\mathcal{R}(\mathcal{U}, \psi)$. As detailed above, by the induction hypothesis, we have that $\langle l_i, \eta_i \rangle \models_{\mathcal{K}} \varphi_2$, and for all $0 \leq j < i$, we have that $\langle l_j, \eta_j \rangle \models \varphi_1$. Hence, as $\langle l_0, \eta_0 \rangle = \langle l, \pi[S := 0] \rangle$, it follows that $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi$, and we are done.

The proof is similar for φ of the form $A\varphi_1 U \varphi_2$, $E\varphi_1 \tilde{U} \varphi_2$, or $A\varphi_1 \tilde{U} \varphi_2$.

- For $\varphi = c.\varphi_1$, we have $f(\varphi, S) = f(\varphi_1, S \cup \{c\})$. By the semantics of TCTL, we have $\langle l, \pi[S := 0] \rangle \models_{\mathcal{R}} \varphi$ iff $\langle l, \pi[S \cup \{c\} := 0] \rangle \models_{\mathcal{R}} \varphi_1$. Since $c \notin bound(\varphi_1)$ and $bound(\varphi_1) \subseteq bound(\varphi)$, then $S \cap bound(\varphi) = \emptyset$ implies that $(S \cup \{c\}) \cap bound(\varphi_1) = \emptyset$. Therefore, by the induction hypothesis, $\langle l, \pi[S \cup \{c\} := 0] \rangle \models_{\mathcal{R}} \varphi_1$ iff $\langle l, \pi[S \cup \{c\} \cup bound(\varphi_1) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S \cup \{c\})$. As $bound(\varphi) = \{c\} \cup bound(\varphi_1)$, this holds iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi_1, S \cup \{c\})$; thus, iff $\langle l, \pi[S \cup bound(\varphi) := 0] \rangle \models_{\mathcal{K}} f(\varphi, S)$, and we are done. ■

Theorems 2 and 3 imply the following.

Corollary 4. *For every timed automaton $\mathcal{U}$ and TCTL formula ψ, the following are equivalent:*

1. $\mathcal{U} \models \psi$.
2. $\mathcal{K}(\mathcal{U}, \psi) \models untime(\psi)$.

The transition from $\mathcal{U}$ to $\mathcal{K}(\mathcal{U}, \psi)$ involves an exponential blow-up, and the translation of ψ into $untime(\psi)$ involves only a linear blow-up. The model-checking problem for CTL can be solved in space that is polynomial in the specification and only poly-logarithmic in the Kripke structure [BVW94]. Corollary 4 then suggests a PSPACE model-checking algorithm for TCTL, matching the known lower bound [ACD93].

3.4 Moving from CTL* to CTL

The formula $fair(S_1, S_2)$ that we use in the definition of $fair$ is not a CTL formula. Moreover, when we use the formula $fair$, it comes before a boolean connective ($\wedge$ or $\rightarrow$) that relates it to formulas of the form $\varphi_1 U \varphi_2$ or $\varphi_1 \tilde{U} \varphi_2$. Hence, as defined now, the function $untime$ translates TCTL formulas to CTL* formulas. In this section we redefine $f(\varphi, S)$ for φ of the form $E\varphi_1 U \varphi_2, A\varphi_1 U \varphi_2, E\varphi_1 \tilde{U} \varphi_2$, or $A\varphi_1 \tilde{U} \varphi_2$, so that the resulting formula will be a CTL formula. Recall that

$$fair(S_1, S_2) = XG(\mathbf{T} \rightarrow (S_1 > 0)) \wedge G(S_2 = 0).$$

Thus, clearly, we could define

$$fair(S_1, S_2) = (S_2 = 0) \wedge [XG((\mathbf{T} \rightarrow (S_1 > 0)) \wedge (S_2 = 0))],$$

which has the form $\xi_1 \wedge XG\xi_2$, where $\xi_1 = (S_2 = 0)$ and $\xi_2 = (\mathbf{T} \rightarrow (S_1 > 0)) \wedge (S_2 = 0)$ are propositional formulas. It follows that we have to translate the following four CTL* formulas to CTL formulas:

1. $E(\xi_1 \wedge XG\xi_2 \wedge \varphi_1 U \varphi_2)$.
2. $A((\xi_1 \wedge XG\xi_2) \rightarrow \varphi_1 U \varphi_2)$.
3. $E(\xi_1 \wedge XG\xi_2 \wedge \varphi_1 \tilde{U} \varphi_2)$.
4. $A((\xi_1 \wedge XG\xi_2) \rightarrow \varphi_1 \tilde{U} \varphi_2)$.

We translate the four formulas to a fragment of CTL* in which the path formulas may contain two temporal operators connected by a boolean operator. Formulas of this fragment have equivalent formulas in CTL [KG96].

1. $E(\xi_1 \wedge XG\xi_2 \wedge \varphi_1 U \varphi_2) = \xi_1 \wedge [(\varphi_2 \wedge EXEG\xi_2) \vee (\varphi_1 \wedge EXE((G\xi_2) \wedge \varphi_1 U \varphi_2)]$.
2. $A((\xi_1 \wedge XG\xi_2) \rightarrow \varphi_1 U \varphi_2) = (\neg\xi_1) \vee \varphi_2 \vee (\varphi_1 \wedge AXA((G\xi_2) \rightarrow \varphi_1 U \varphi_2)) \vee AXAF\neg\xi_2$.
3. $E(\xi_1 \wedge XG\xi_2 \wedge \varphi_1 \tilde{U} \varphi_2) = \xi_1 \wedge [(\varphi_1 \wedge \varphi_2 \wedge EXEG\xi_2) \vee (\varphi_2 \wedge EXE((G\xi_2) \wedge \varphi_1 \tilde{U} \varphi_2))]$.
4. $A((\xi_1 \wedge XG\xi_2) \rightarrow \varphi_1 \tilde{U} \varphi_2) = (\neg\xi_1) \vee (\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge AXA((G\xi_2) \rightarrow \varphi_1 \tilde{U} \varphi_2)) \vee AXAF\neg\xi_2$.

Finally, as the formula $EXEG\xi_2$ is valid in $K(\mathcal{U}, \psi)$, we replace it with **true** and replace its negation $AXAF\neg\xi_2$ with **false**. Accordingly, we now have.

1. $f(E\varphi_1 U \varphi_2, S) = \xi_1 \wedge [\varphi_2 \vee (\varphi_1 \wedge EXE((G\xi_2) \wedge \varphi_1 U \varphi_2))]$.
2. $f(A\varphi_1 U \varphi_2, S) = (\neg\xi_1) \vee \varphi_2 \vee (\varphi_1 \wedge AXA((G\xi_2) \rightarrow \varphi_1 U \varphi_2))$.
3. $f(E\varphi_1 \tilde{U} \varphi_2, S) = \xi_1 \wedge [(\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge EXE((G\xi_2) \wedge \varphi_1 \tilde{U} \varphi_2))]$.
4. $f(A\varphi_1 \tilde{U} \varphi_2, S) = (\neg\xi_1) \vee (\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge AXA((G\xi_2) \rightarrow \varphi_1 \tilde{U} \varphi_2))$.

This completes the translation of $untime(\psi)$ into CTL.

4 Discussion

In this paper we suggested a reduction from TCTL model checking to CTL model checking. Recall that the way we define the semantics for TCTL, we do not require path quantification to range only over paths for which time diverges. Since we can replace the divergence requirement by a fairness constraint on $\mathcal{K}(\mathcal{U}, \psi)$ (see [HKV96]), it is easy to extend our algorithm to handle a semantics in which path quantification ranges only over divergent paths. Then, TCTL model checking is reduced to Fair-CTL model checking. By [KV95], the latter can be solved with the same space complexity as CTL model checking. Hence, the PSPACE complexity is preserved.

Our reduction handles the reset quantifier of TCTL by augmenting the region graph induced by a timed automaton with new transitions and limiting path quantification in the formula. As such, our reduction can be easily adjusted to handle model checking of TCTL formulas when interpreted with respect to hybrid systems with finite bisimulations [Hen95].

The advantage of the algorithm that follows from our reduction is the existence of fine-tuned tools for CTL model checking. The algorithm can be optimized further by exploiting the special structure of $\mathcal{K}(\mathcal{U}, \psi)$. For example, the optimization suggested in [HKV96], which integrates states that differ only in their region element into a single state, can be used also here. It remains to be seen how the algorithm performs in practice.

References

[ACD93] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–236, 1994.

[AH92] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 74–106. 1992.

[AH93] R. Alur and T. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, May 1993.

[AH94] R. Alur and T.A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

[BHSV+96] R.K. Brayton, G.D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. Edwards, S. Khatri, T. Kukimoto, A. Pardo, S. Qadeer, R.K. Ranjan, S. Sarwary, T.R. Shiple, G. Swamy, and T. Villa. VIS: a system for verification and synthesis. In *Computer Aided Verification, Proc. 8th Int. Workshop*, volume 1102 of *Lecture Notes in Computer Science*, pages 428–432. Springer-Verlag, 1996.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Computer Aided Verification, Proc. 6th Int. Conference*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, 1994.

[CCM+94] S. Campos, E.M. Clarke, W. Marrero, M. Minea, and H. Hiraishi. Computing quantitative characteristics of finite-state real-time systems. In *Proceedings of the 15th Annual Real-time Systems Symposium*. IEEE Computer Society Press, 1994.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[CGL93] E.M. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Decade of Concurrency—Reflections and Perspectives (Proceedings of REX School)*, Lecture Notes in Computer Science, pages 124–175. Springer-Verlag, 1993.

[EMSS90] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *Proc. 2nd Workshop on Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 136–145. Springer-Verlag, 1990.

[FGK+96] J.-C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu. CADP: a protocol validitation and verification toolbox. In *Computer Aided Verification, Proc. 8th Int. Workshop*, volume 1102 of *Lecture Notes in Computer Science*, pages 437–440. Springer-Verlag, 1996.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[HKV96] T.A. Henzinger, O. Kupferman, and M.Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Proc. 7th Conference on Concurrency Theory*, Pisa, August 1996. Springer-Verlag.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *ICALP 92: Automata, Languages, and Programming*, Lecture Notes in Computer Science 623, pages 545–558. Springer-Verlag, 1992.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[KG96] O. Kupferman and O. Grumberg. Buy one, get one free! *Journal of Logic and Computation*, 6(4), 1996.

[KV95] O. Kupferman and M.Y. Vardi. On the complexity of branching modular model checking. In *Proc. 6th Conference on Concurrency Theory*, pages 408–422, Philadelphia, August 1995. Springer-Verlag.

[LL95] F. Laroussinie and K. G. Larsen. Compositional model checking of real time systems. In *Proc. 6th Conference on Concurrency Theory*, pages 27–41, Philadelphia, August 1995. Springer-Verlag.

[McM93] K.L. McMillan. *Symbolic model checking*. Kluwer Academic Publishers, 1993.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[QS81] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, volume 137, pages 337–351. Springer-Verlag, Lecture Notes in Computer Science, 1981.

[SS95] O.V. Sokolsky and S.A. Smolka. Local model checking for real-time systems. In *Computer Aided Verification, Proc. 7th Int. Workshop*, Lecture Notes in Computer Science 939, pages 211–224, Liege, July 1995.