It's About Time: Real-time Logics Reviewed[⋆ ⋆⋆]

Thomas A. Henzinger

Electrical Engineering & Computer Sciences
University of California at Berkeley

tah@eecs.berkeley.edu
www.eecs.berkeley.edu/~tah

Abstract. We summarize and reorganize some of the last decade's research on real-time extensions of temporal logic. Our main focus is on tableau constructions for model checking linear temporal formulas with timing constraints. In particular, we find that a great deal of real-time verification can be performed in polynomial space, but also that considerable care must be exercised in order to keep the real-time verification problem in polynomial space, or even decidable.

1 Introduction

The execution of a reactive system results in an infinite sequence of observations. Requirements on execution sequences can be specified in (linear) temporal logic.[3] The model-checking problem asks, given a reactive system and a temporal formula, if all execution sequences of the system satisfy the formula.

Temporal logic is a popular specification language for two reasons. First, temporal logic is *reasonably expressive*. In practice, temporal logic allows the specification of important requirements such as invariance and response [MP92]. In theory, the expressive power of temporal logic is robust: temporal logic is as expressive as a certain first-order monadic logic on the natural numbers [GPSS80],

⋆ An abbreviated version of this paper appeared in the *Proceedings of the Tenth International Conference on Concurrency Theory* (CONCUR), Lecture Notes in Computer Science 1466, Springer-Verlag, 1998, pp. 439–454.

⋆⋆ This research was supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grant CCR-9504469, by the Defense Advanced Research Projects Agency grant NAG2-1214, by the Army Research Office MURI grant DAAH-04-96-1-0341, and by the Semiconductor Research Corporation contract 97-DC-324.041.

3 There are also other, branching varieties of temporal logic, for specifying requirements on execution trees [Eme90]. In this paper, we are solely concerned with the linear view. For model checking with integer-time branching temporal logics, see [EMSS90, Eme92]; for model checking with real-time branching temporal logics, see [ACD93, HNSY94].

and with the addition of a (second-order) hiding operator, as expressive as Büchi automata [Büc62].

Second, temporal logic is *reasonably efficient.* In practice, model checkers have been successful both in hardware and protocol design [CK96]. In theory, the complexity of temporal logic is not dominant: if a reactive system is given as a product of Büchi automata, the model-checking problem can be solved in polynomial space [LP85], and thus is no harder than the most basic of verification problems —invariance checking.[4] The efficiency of temporal logic is due to a careful choice of operators. For example, the addition of the hiding operator would cause an exponential increase in the complexity of model checking [Sis83].

We illustrate that with a careful choice of operators, both pleasing properties of temporal logic —reasonable, robust expressiveness and reasonable, polynomial-space efficiency— can be maintained when moving from reactive to real-time systems.

The execution of a real-time system results in an infinite sequence of observations that are time-stamped with reals. A paradigmatic language for describing real-time systems is obtained by considering Alur-Dill automata with parallel composition [AD94]. Invariance checking for products of Alur-Dill automata can still be performed in polynomial space. We present several operators that refer to time stamps but can be added to temporal logic without increasing the polynomial-space complexity of model checking. The careful choice of real-time operators is even more critical than in the reactive case, because a wrong choice can easily render the model-checking problem undecidable.

In Section 2, we review in some detail the properties of temporal logic that are relevant to this discussion. In Section 3, we classify integer-time operators into operators that can be model checked in polynomial space, and operators that demand exponential space. In Section 4, we see that when interpreted over real time, the first class of operators can still be model checked in polynomial space, while the second class of operators becomes undecidable. The polynomial-space real-time operators we advocate are also expressively robust: the extended temporal logics are as expressive as a certain first-order monadic logic on the reals, and with the addition of hiding, as expressive as Alur-Dill automata.

2 Temporal Logic

Let Π be a set of propositions. An *observation* is a mapping from Π to the set of truth values. A *trace* $\sigma = s_0 s_1 s_2 \ldots$ is an infinite sequence of observations. The *positions* of σ are the nonnegative integers. For a position p of σ, the *observation of* σ *at position* p is denoted $\sigma[p] = s_p$. The *temporal formulas* are defined by the grammar

$$\phi \ ::= \ \pi \mid \phi \vee \phi \mid \neg\phi \mid \phi\,\mathcal{U}\phi$$

[4] Invariance checking on a single Büchi automaton can, of course, be performed in linear time. But a language without parallel composition (product) is not useful for describing nontrivial reactive systems.

where π is a proposition from Π. The temporal operator $\mathcal{U}$ is pronounced "until." Given a position p of a trace σ, and a temporal formula ϕ, the relation $(\sigma, p) \models \phi$, pronounced "the formula ϕ is true at position p of trace σ," is defined inductively:

$(\sigma, p) \models \pi$ if $\sigma[p](\pi) = \mathit{true}$;
$(\sigma, p) \models \phi_1 \vee \phi_2$ if $(\sigma, p) \models \phi_1$ or $(\sigma, p) \models \phi_2$;
$(\sigma, p) \models \neg\phi$ if not $(\sigma, p) \models \phi$;
$(\sigma, p) \models \phi_1 \mathcal{U} \phi_2$ if there exists a position $r > p$ of σ such that $(\sigma, r) \models \phi_2$, and for all positions q of σ, if $p < q < r$ then $(\sigma, q) \models \phi_1$.[5]

The trace σ *satisfies* the temporal formula ϕ if $(\sigma, 0) \models \phi$. The temporal formula ϕ *defines* the set of traces that satisfy ϕ.

Useful defined operators are $\bigcirc$ ("next"), $\Diamond$ ("eventually in the future"), and $\Box$ ("always in the future"): $\bigcirc\phi = \mathit{false}\,\mathcal{U}\phi$, $\Diamond\phi = \mathit{true}\,\mathcal{U}\phi$, and $\Box\phi = \neg\Diamond\neg\phi$. For example, the response formula $\Box(a \to \Diamond b)$ asserts that every observation of a is followed by an observation of b.

Satisfiability. In order to solve the model-checking problem, it is useful to study the *satisfiability problem* for temporal logic: given a temporal formula ϕ, is there a trace that satisfies ϕ? The satisfiability problem can be solved by constructing a Büchi automaton B_ϕ, called the *tableau* of ϕ, which accepts precisely the traces that satisfy ϕ [Wol82]. Then, ϕ is satisfiable iff B_ϕ is nonempty.

In this method, it is the size of the tableau B_ϕ which determines the efficiency of solving the satisfiability problem. Every location ℓ of the Büchi automaton B_ϕ is a set of subformulas of ϕ. When an execution of B_ϕ visits location ℓ, the subformulas in ℓ represent constraints on the remainder of the input trace, which must be satisfied in order for the automaton to accept. The tableau construction is possible, because all temporal constraints can be propagated from one location to the next: if ℓ contains the until formula $\psi_1 \mathcal{U} \psi_2$, then each successor location of ℓ must contain either ψ_2, or both ψ_1 and $\psi_1 \mathcal{U} \psi_2$. Since the number of subformulas of ϕ is linear in the length of ϕ, the number of locations of B_ϕ is exponential in ϕ. Since Büchi emptiness can be checked in space polylogarithmic in the number of locations [Sav70, VW94], the satisfiability problem for temporal logic can be solved in polynomial space.

This is also a lower bound [SC85]. Given a polynomial f and a Turing machine M that uses space $f(n)$ for inputs of size n, we can construct a temporal formula ϕ_M of length polynomial in n which is satisfied precisely by the traces that do not encode accepting computations of M. Then, M has an accepting computation iff $\neg\phi_M$ is satisfiable. The key to the construction of ϕ_M is the formula $\Diamond(p \wedge \bigcirc \cdots \bigcirc \neg p)$, with $f(n)$ next operators. Formulas of this form can be used to encode the fact that the contents of one of the $f(n)$ tape cells is not properly maintained from one configuration of M to the next.

[5] The strictness of the until operator, which does not constrain the current state, will facilitate the move to real time: in real time, strict until cannot be defined from weak until and next [Ras98].

Model checking. A finite-state reactive system is naturally described by a product $B_S = B_1 \times \cdots \times B_m$ of Büchi automata, which represent the state-transition graphs and the fairness assumptions of the individual system components. This leads to the following formulation of the *model-checking problem*: given a product B_S of Büchi automata and a temporal formula ϕ, do all traces that are accepted by B_S satisfy ϕ? The model-checking problem can be solved in polynomial space, by checking the emptiness of the product automaton $B_S \times B_{\neg\phi}$ [LP85]. This is again a lower bound [SC85]: the model-checking problem is no simpler than the satisfiability problem, because a temporal formula ϕ is unsatisfiable iff the negated formula $\neg\phi$ is satisfied by all infinite paths of the complete observation graph.

Expressiveness. The expressive power of temporal logic is closely related to Büchi automata. A trace set L is *ω-regular* if there is a Büchi automaton that accepts precisely the traces in L. The tableau construction shows that if a trace set L is definable by a temporal formula, then L is ω-regular. The converse is not necessarily true, and the gap between temporal definability and ω-regularity can be closed in several ways [Tho90] —for example, via the operation of hiding propositions, which is an important operation in specification. A *temporal formula with hidden propositions* is has the form $(\exists \pi_1, \ldots, \pi_n)\phi$, where $\pi_1, \ldots, \pi_n$ are propositions and ϕ is a temporal formula. The semantics of existential quantification is standard: $(\sigma, p) \models (\exists \pi)\phi$ if $(\sigma', p) \models \phi$ for some trace σ' that differs from σ only in the values that are given to π by the observations of σ'.

Given a Büchi automaton B with the set $\{\pi_1, \ldots, \pi_n\}$ of locations, we can construct a temporal formula ϕ_B with the hidden propositions $\pi_1, \ldots, \pi_n$ such that ϕ_B defines the set of traces that are accepted by B. Thus, a trace set L is definable by a temporal formula with hidden propositions iff L is ω-regular.

Temporal logic with hidden propositions, however, is an expensive specification language. Since the formula $(\exists \pi)\phi$ is satisfiable iff ϕ is satisfiable, the satisfiability problem for temporal logic with hidden propositions can still be solved in polynomial space. But the reduction from model checking to satisfiability involves negation, and the temporal formulas with hidden propositions are not closed under negation. The model-checking problem for a Büchi system B_S and a formula $(\exists \pi_1, \ldots, \pi_n)\phi$ requires exponential space, by checking the emptiness of the automaton $B_S \times \neg(\exists \pi_1, \ldots, \pi_n)B_\phi$, whose construction involves the complementation of a Büchi automaton [Saf88]. This is also a lower bound [Sis83]. For the hidden propositions $\pi_1, \ldots, \pi_n$, we can assert by a formula whose length is polynomial in n that each proposition π_i is true precisely at all positions that are multiples of 2^i. In this way, we can define by a formula whose length is polynomial in n the traces that do not encode accepting computations of a Turing machine that uses space 2^n.

3 Integer-time Logics

Temporal logic is good for specifying qualitative requirements on execution sequences, such as response, but inconvenient for specifying quantitative requirements, such as bounded response. A bounded-response requirement may assert that every observation of a is followed within 5 positions by an observation of b. This requirement can be specified, rather awkwardly, by the temporal formula

$$\Box(a \rightarrow \bigcirc(b \vee \bigcirc(b \vee \bigcirc(b \vee \bigcirc(b \vee \bigcirc b))))). \qquad (\dagger)$$

In order to facilitate more succinct specifications of quantitative requirements, we can annotate temporal operators with quantitative constraints.

Constrained temporal operators. The *constrained temporal formulas* are defined by the grammar

$$\phi \ ::= \ \pi \mid \phi \vee \phi \mid \neg\phi \mid \bigcirc\phi \mid \phi\, \mathcal{U}_{\sim c}\, \phi \mid \ \rhd_I \phi$$

where π is a proposition, $\sim$ is an inequality operator from the set $\{<, \leq, \geq, >\}$, the constant c is a nonnegative integer, and I is an interval with integer endpoints. For our purposes, an *interval* is a convex subset of the nonnegative reals —intervals can be open, half-open, or closed; bounded or unbounded. For an interval I and a nonnegative real p, we freely use notation such as $p + I$ for the interval $\{p + q \mid q \in I\}$, and $I \sim p$ for the condition that $q \sim p$ for all reals $q \in I$. An *overconstrained temporal formula* is a constrained temporal formula that may contain also subformulas of the form $\phi_1 \mathcal{U}_{=c}\, \phi_2$.

The truth of an (over)constrained temporal formula ϕ at position p of a trace σ is defined as follows:

$(\sigma, p) \models \phi_1 \mathcal{U}_{\sim c}\, \phi_2$ if there exists a position $r > p$ of σ with $r \sim (p + c)$ such that $(\sigma, r) \models \phi_2$, and for all positions q of σ, if $p < q < r$ then $(\sigma, q) \models \phi_1$;

$(\sigma, p) \models \ \rhd_I \phi$ if there exists a position $r \in (p + I)$ of σ such that $(\sigma, r) \models \phi$, and for all positions q of σ, if $p < q < r$ then $(\sigma, q) \models \neg\phi$.

The temporal operator $\rhd$, which is pronounced "earliest," was introduced in [RS97], motivated by the event-predicting clocks of [AFH94]. In words, the formula $\rhd_I \phi$ is true at position p iff there is a later position $r > p$ at which ϕ is true, and the earliest such position lies in the interval $p + I$.

Useful defined operators are unconstrained until, constrained eventually, and constrained always: $\phi_1 \mathcal{U} \phi_2 = \phi_1 \mathcal{U}_{>0}\, \phi_2$, $\Diamond_{\sim c}\, \phi = \mathit{true}\, \mathcal{U}_{\sim c}\, \phi$, and $\Box_{\sim c}\, \phi = \neg\Diamond_{\sim c} \neg\phi$. For example, the bounded-response requirement ($\dagger$) can be specified by the formula $\Box(a \rightarrow \Diamond_{\leq 5}\, b)$, or alternatively, by $\Box(a \rightarrow \ \rhd_{(0,5]}\, b)$. While these two formulas are equivalent, it is important to notice the difference between, say, the requirements $\Diamond_{\geq 5}\, b$ and $\rhd_{[5,\infty)}\, b$: the formula $\Diamond_{\geq 5}\, b$ asserts that *some* observation of b will occur after 5 positions or later; the formula $\rhd_{[5,\infty)}\, b$ makes the stronger assertion that *the earliest* observation of b will occur after 5 positions or later.

Model checking. Constrained and overconstrained temporal operators do not add expressive power to temporal logic, only succinctness. While for the overconstrained temporal formulas, the exponential increase in succinctness comes at an exponential cost in efficiency, for the constrained temporal formulas the exponential increase in succinctness comes at no cost: constrained operators can be model checked in polynomial space; overconstrained operators cannot.

In the tableau construction, suppose that location ℓ contains the constrained until formula $\psi_1 \,\mathcal{U}_{\leq 5}\, \psi_2$. Then each successor location of ℓ must contain either ψ_2, or both ψ_1 and $\psi_1 \,\mathcal{U}_{\leq 4} \psi_2$. If ℓ contains $\psi_1 \,\mathcal{U}_{\geq 5}\, \psi_2$, then each successor location of ℓ must contain both ψ_1 and $\psi_1 \,\mathcal{U}_{\geq 4} \psi_2$. If ℓ contains the earliest formula $\rhd_{[5,5]}\, \psi$, then each successor location of ℓ must contain both the complement[6] of ψ and $\rhd_{[4,4]}\, \psi$. Finally, if ℓ contains the overconstrained formula $\psi_1 \,\mathcal{U}_{=5}\, \psi_2$, then each successor location of ℓ must contain both ψ_1 and $\psi_1 \,\mathcal{U}_{=4}\, \psi_2$. Hence, the formula $\psi_1 \,\mathcal{U}_{\sim(c-1)}\, \psi_2$ has to be declared a "subformula" of $\psi_1 \,\mathcal{U}_{\sim c}\, \psi_2$, and the formula $\rhd_{I-1}\, \psi$ has to be declared a "subformula" of $\rhd_I\, \psi$. This, however, makes the number of subformulas of a given formula ϕ exponential in the length of ϕ.[7] It follows that the number of locations of the tableau B_ϕ is doubly exponential in ϕ, and the model-checking procedure for the overconstrained temporal formulas requires exponential space.

For the constrained temporal formulas, a powerful optimization is possible (as discussed for branching time in [EMSS90]). For given subformulas ψ_1 and ψ_2 of ϕ, a location of B_ϕ needs to contain at most one formula of the form $\psi_1 \,\mathcal{U}_{\sim c}\, \psi_2$. This is because the conjunction of $\psi_1 \,\mathcal{U}_{\sim c_1}\, \psi_2$ and $\psi_1 \,\mathcal{U}_{\sim c_2}\, \psi_2$ is equivalent to the single constraint $\psi_1 \,\mathcal{U}_{\sim \min(c_1,c_2)}\, \psi_2$ if $\sim \in \{<, \leq\}$, and to $\psi_1 \,\mathcal{U}_{\sim \max(c_1,c_2)}\, \psi_2$ if $\sim \in \{\geq, >\}$. Similarly, for a given subformula ψ, a location of B_ϕ needs to contain at most one formula of the form $\rhd_I\, \psi$. This is because the conjunction of $\rhd_{I_1}\, \psi$ and $\rhd_{I_2}\, \psi$ is equivalent to the single constraint $\rhd_{I_1 \cap I_2}\, \psi$. Hence, if ϕ has length n and contains no integer constants greater than c, then the number of locations of the tableau B_ϕ can be reduced to $2^{O(n \log c)}$. This gives a polynomial-space model-checking procedure for the constrained temporal formulas.

The optimization is impossible for overconstrained formulas of the form $\Diamond_{=c}\, \psi$, which are not closed under conjunction. Indeed, for every Turing machine M that uses space 2^n, we can construct an overconstrained temporal formula ϕ_M of length polynomial in n which is satisfied precisely by the traces that do not encode accepting computations of M [AH94]. The key to the construction of ϕ_M is the formula $\Diamond(p \wedge \Diamond_{=2^n} \neg p)$, whose length is linear in n. By reducing satisfiability to model checking as in the unconstrained case, we conclude that exponential space is a lower bound for model checking overconstrained temporal formulas.

Alur-Dill automata. If quantitative behavior is of interest, it is convenient to

[6] In this expository paper, we provide no complete definitions for tableau constructions, but only the key ideas behind the constructions. In particular, we leave it to the reader to define the complement of an (over)constrained temporal formula. This can be done, for example, by introducing duals of the constrained until and earliest operators that allow all negations to be pushed to the front of propositions [AFH96].

[7] If the integer constants that occur in (over)constrained temporal operators are written in unary notation, then the number of subformulas of ϕ remains linear in the length of ϕ, and all overconstrained temporal formulas can be model checked in polynomial space.

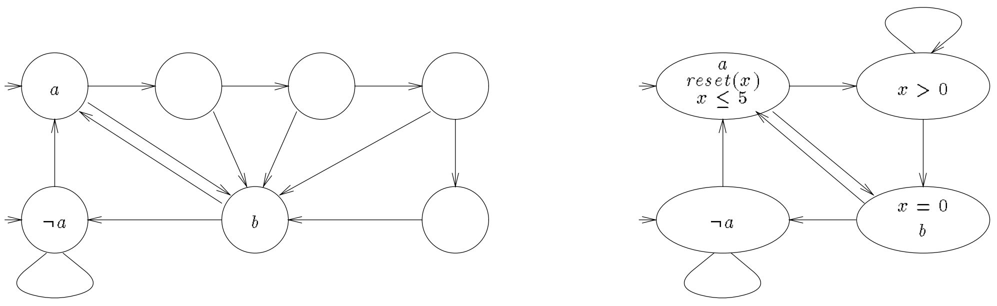

Fig. 1. Bounded response

have succinct languages not only for requirement specification but also for system description. A time-constrained system is naturally described by a product $A_S = A_1 \times \cdots \times A_m$ of Alur-Dill automata [AD94]. An *Alur-Dill automaton* $A = (B, X, \alpha, \beta, \gamma)$ consists of a Büchi automaton B, a finite set X of clocks, and three labeling functions on the locations of B. The exit-guard function α and the entry-guard function β each map every location of B to a finite sets of clock constraints, and the reset function γ maps every location of B to a set of clocks from X. A *clock constraint* is an inequality of the form $x \sim d$, where $x \in X$ is a clock, $\sim \in \{<, \leq, =, \geq, >\}$, and d is a nonnegative integer constant.[8]

Under the assumption that consecutive observations in a trace are separated by exactly 1 time unit, every Alur-Dill automaton accepts a set of traces. In this case, the clocks behave like integer variables. If $(x \sim d) \in \alpha(\ell)$, then the execution of A can exit location ℓ if the value of x satisfies the constraint $x \sim d$. Then 1 time unit expires, and all clock values *decrease* by 1. If $(x \sim d) \in \beta(\ell')$, then the execution of A can enter the next location ℓ' if the (decreased) value of x satisfies the constraint $x \sim d$. If $x \in \gamma(\ell')$, then once the the execution of A has entered location ℓ', the value of the clock x can change nondeterministically to *any nonnegative* integer.[9] For example, Figure 1 shows a (rather awkward) Büchi automaton and a (more succinct) Alur-Dill automaton, both of which accept precisely the traces that satisfy the bounded-response requirement (†). Neither automaton has acceptance conditions, and it is assumed that a and b are mutually exclusive.

Formally, the Alur-Dill automaton A *accepts* the trace σ if (1) the underlying Büchi automaton B accepts σ, along some infinite execution that visits the sequence $\ell_0\ell_1\ell_2\ldots$ of locations, and (2) for each clock $x \in X$, there is an infinite sequence $u_0v_1u_1v_2u_2v_3\ldots$ of (possibly negative) integer clock values such that

[8] It is convenient to label locations, rather than transitions, with clock constraints, just like in Büchi tableaux it is convenient to label locations, not transitions, with input observations.

[9] Our choice of decreasing clocks with nondeterministic resets, rather than increasing clocks with resets to 0, will simplify later tableau constructions.

for all $i \geq 0$,

(2a) [exit guard] if $(x \sim d) \in \alpha(\ell_i)$ then $u_i \sim d$,
(2b) [clock progress] $v_{i+1} = u_i - 1$,
(2c) [entry guard] if $(x \sim d) \in \beta(\ell_{i+1})$ then $v_{i+1} \sim d$, and
(2d) [clock reset] if $x \notin \gamma(\ell_{i+1})$ then $u_{i+1} = v_{i+1}$ else $u_{i+1} \geq 0$.

Every Alur-Dill automaton A can be translated into a Büchi automaton $\mathcal{B}(A)$ that accepts the same traces. If A has l locations, k clocks, and contains no integer constants greater than d, then $\mathcal{B}(A)$ has $l \cdot 2^{O(k \log d)}$ locations: each location of $\mathcal{B}(A)$ consists of a location of A and a k-vector of integers between -1 and $d+1$, which represent clock values (the exact value of a clock is immaterial if it is less than 0 or greater than d) [AH92b]. Hence, the emptiness problem for Alur-Dill automata can be solved in polynomial space.

Integer-time model checking. This leads to the following formulation of the *integer-time model-checking problem*: given a product $A_S = A_1 \times \cdots \times A_m$ of Alur-Dill automata and an (over)constrained temporal formula ϕ, do all traces that are accepted by A_S satisfy ϕ? We can solve the integer-time model-checking problem by first constructing a a Büchi system $\mathcal{B}(A_S)$ and a Büchi tableau $B_{\neg\phi}$, and then checking the emptiness of $\mathcal{B}(A_S) \times B_{\neg\phi}$. Notice that if each component automaton A_i has l_i locations, k_i clocks, and contains no integer constants greater than d_i, then the product automaton A_S has $\prod_{i=1}^{m} l_i$ locations, $\sum_{i=1}^{m} k_i$ clocks, and contains no constant greater than $\max_{i=1}^{m} d_i$ [AD94]. It follows that the integer-time model-checking problem can be solved in polynomial space for the constrained temporal formulas (and in exponential space for the overconstrained formulas).

Clock tableaux. There is an alternative approach to integer-time model checking for constrained formulas: we can first construct a suitable Alur-Dill automaton $A_{\neg\phi}$ and then check the emptiness of the Büchi automaton $\mathcal{B}(A_S \times A_{\neg\phi})$ [AFH96]. Given a constrained temporal formula ϕ, the *clock tableau* A_ϕ is an Alur-Dill automaton that accepts precisely the traces that satisfy ϕ. For each syntactic subformula ψ of ϕ, we define two copies —the *initial* copy ψ^I, and the *stale* copy ψ^S— and a clock x_ψ. Every location of the clock tableau A_ϕ is a set of initial and stale copies of subformulas of ϕ. The initial copy ψ^I indicates that the remainder of the input trace must satisfy ψ. The stale copy ψ^S, which is propagated from location to location, indicates that the remainder of the input trace need not satisfy ψ itself but an inherited constraint that also depends on the current value of the clock x_ψ.

The transition relation and guard functions of the clock tableau A_ϕ are defined by the following rules. If location ℓ contains the initial copy ψ^I of the constrained until formula $\psi = (\psi_1 \,\mathcal{U}_{\sim c}\, \psi_2)$, then $(x_\psi \sim c) \in \alpha(\ell)$ and each successor location of ℓ contains the stale copy ψ^S. If ℓ contains the stale copy ψ^S, then either ℓ contains the initial copy ψ_2^I and $(x_\psi = 0) \in \beta(\ell)$, or ℓ contains the initial copy ψ_1^I and each successor location of ℓ contains the stale copy ψ^S. If ℓ contains the initial copy ψ^I of the earliest formula $\psi = (\rhd_I \psi_0)$, then $(x_\psi \in I) \in \alpha(\ell)$[10] and

[10] If the interval I is bounded from both below and above, then the condition $x_\psi \in I$

each successor of ℓ contains the stale copy ψ^S. If ℓ contains the stale copy ψ^S, then either ℓ contains the initial copy ψ_0^I and $(x_\psi = 0) \in \beta(\ell)$, or ℓ contains the initial copy of the complement of ψ_0 and each successor location of ℓ contains the stale copy ψ^S.

The reset function γ of the clock tableau A_ϕ is defined as follows. For constrained until formulas $\psi = (\psi_1 \mathcal{U}_{\sim c} \psi_2)$ with $\sim \in \{\geq, >\}$, we require that $x_\psi \in \gamma(\ell)$ iff ℓ contains the initial copy ψ^I. For constrained until formulas $\psi = (\psi_1 \mathcal{U}_{\sim c} \psi_2)$ with $\sim \in \{<, \leq\}$, as well as for earliest formulas $\psi = (\rhd_I \psi_0)$, we require that $x_\psi \in \gamma(\ell)$ iff (1) ℓ contains the initial copy ψ^I and (2) if ℓ also contains the stale copy ψ^S, then $(x_\psi = 0) \in \beta(\ell)$. These conditions are motivated by the arguments for the optimization of the Büchi-tableau construction given above.

In summary, if the constrained temporal formula ϕ has length n, then the clock tableau A_ϕ has $2^{O(n)}$ locations, $O(n)$ clocks, and contains no integer constants greater than the largest constant in ϕ. Together with the product construction and emptiness check for Alur-Dill automata, this gives again a polynomial-space algorithm for integer-time model checking of constrained temporal formulas.

4 Real-time Logics

While integer-time models assume that consecutive observations in a trace are separated by exactly 1 time unit, real-time models admit arbitrary real-numbered delays between observations. A *timing* $\tau = t_1 t_2 t_3 \ldots$ is an infinite sequence of positive reals whose sum diverges: $\sum_{i \geq 1} t_i = \infty$. A *timed trace* (σ, τ) consists of a trace σ and a timing τ. Each real t_i represents the delay between the $(i-1)$-st and i-th observation of σ. The *positions* of (σ, τ) are the nonnegative reals in the set $\{\sum_{i=1}^{k} t_i \mid k \geq 0\}$, which represent the times at which observations occur. For a position p of the timed trace (σ, τ), the *observation of* (σ, τ) *at position* p is defined by $(\sigma, \tau)[p] = s_k$ if $p = \sum_{i=1}^{k} t_i$ (in particular, $(\sigma, \tau)[0] = s_0$).

The (over)constrained temporal formulas can be interpreted over timed traces, simply by replacing the trace σ in the definition of the relation $(\sigma, p) \models \phi$ by a timed trace (σ, τ). For example, the constrained temporal formula $a \wedge \Box(a \rightarrow \rhd_{[1,1]} a)$ asserts that the proposition a is true exactly at the integer points in real time.

Alur-Dill automata can also be interpreted over timed traces. The Alur-Dill automaton A *accepts* the timed trace (σ, τ) if in the definition of trace acceptance, the sequence of integer clock values is replaced by a sequence of real clock values, and the clock-progress condition (2b) is replaced by the condition $v_{i+1} = u_i - t_{i+1}$. We say that the Alur-Dill automaton A *defines* the set of timed traces that are accepted by A. The *real-time emptiness problem* for Alur-Dill automata asks if a given automaton accepts any timed trace; the *real-time universality problem* asks if a given automaton accepts every timed trace.

gives rise to two clock constraints.

Every Alur-Dill automaton A can be translated into a Büchi automaton $\mathcal{R}(A)$, called the *region automaton* of A, such that $\mathcal{R}(A)$ accepts a trace σ iff there is a timing τ so that A accepts the timed trace (σ, τ). If A has l locations, k clocks, and contains no integer constants greater than d, then $\mathcal{R}(A)$ has $l \cdot 2^{O(k \log d)} \cdot O(2^k k!)$ locations: each location of $\mathcal{R}(A)$ consists of a location of A, a k-vector of integers between -1 and $d+1$, which represent the integer parts of the clock values, and an ordered partition of the clocks, which represents the ordering of the fractional parts of the clock values [AD94]. It follows that the real-time emptiness problem for Alur-Dill automata can be solved in polynomial space.

This leads to the following formulation of the *real-time model-checking problem*: given a product $A_S = A_1 \times \cdots \times A_m$ of Alur-Dill automata and an (over)constrained temporal formula ϕ, do all timed traces that are accepted by A_S satisfy ϕ? For the constrained temporal formulas, the real-time model-checking problem can be solved using the clock-tableau approach: first construct the clock tableau $A_{\neg\phi}$ (which is defined exactly as in the case of integer time) and then check the emptiness of the region automaton $\mathcal{R}(A_S \times A_{\neg\phi})$. This algorithm uses polynomial space.

Theorem 1. *[AFH96, RS97] The real-time model-checking problem for the constrained temporal formulas can be solved in polynomial space.*

In the case of real time, the restriction to constrained temporal formulas is essential: for the overconstrained formulas, where the integer-time model-checking problem requires exponential space, the real-time model-checking problem is undecidable. Given an arbitrary Turing machine M, we can construct an overconstrained temporal formula ϕ_M that is satisfied precisely by the timed traces that do not encode accepting computations of M [AH94]. In real time, any finite number of observations may occur within a single time unit. In this way, Turing machine configurations of arbitrary length can be encoded within a time interval of length 1. Then, the overconstrained formula $\Diamond(p \wedge \Diamond_{=1} \neg p)$ encodes the fact that the contents of a tape cell is not properly maintained from one configuration of M to the next.

Theorem 2. *[AH93, AH94] The real-time model-checking problem for the overconstrained temporal formulas is undecidable. (The integer-time model-checking problem for the overconstrained temporal formulas is complete for exponential space.)*

The timed trace set that is defined by the overconstrained formula $\Diamond(p \wedge \Diamond_{=1} \neg p)$ can also be defined by an Alur-Dill automaton. It follows that the real-time universality problem for Alur-Dill automata is undecidable [AD94].[11]

Recall that the addition of hiding to temporal logic allows the definition of all ω-regular trace sets. Analogously, the extension of the constrained temporal formulas with hidden propositions allows the definition of all timed trace sets that

[11] This does not contradict the decidability of the real-time emptiness problem, because unlike temporal formulas, Alur-Dill automata are not closed under complementation.

are definable by Alur-Dill automata. This is because every Alur-Dill automaton A can be translated into a constrained temporal formula ϕ_A with hidden propositions —one for each location, one for each clock constraint, and one for each reset action of A— such that A and ϕ_A define the same timed trace set. Since the universality problem for Alur-Dill automata is undecidable, it follows that hiding renders the real-time model-checking problem for the constrained temporal formulas undecidable. This is in contrast to the case of integer time, where hiding increases the model-checking complexity for the constrained temporal formulas by an exponential.

Theorem 3. *[Wil94, HRS98] A set of timed traces is definable by an Alur-Dill automaton iff it is definable by a constrained temporal formula with hidden propositions.*

Corollary 4. *The real-time model-checking problem for the constrained temporal formulas with hidden propositions is undecidable. (The integer-time model-checking problem for the constrained temporal formulas with hidden propositions is complete for exponential space.)*

We have seen that both overconstraining and hiding render the real-time model-checking problem undecidable. We conclude with two semantic extensions and a syntactic extension of real-time logics which are benign: superdense and interleaving models of time, and constrained temporal operators that refer to the past, rather than the future. Two more extensions of syntax —interval-constrained until operators in the case of real time, and clock constraints for integer time— have exponential cost.

Superdense models. In a timed trace, there are "gaps" between observations, which occur at discrete *points* in real time. Alternatively, we can define a timed trace as a function from the nonnegative reals to the observations [BKP86]. In this view, observations have duration, and correspond to *intervals* in real time. Without loss of generality, we assume that every even-numbered observation corresponds to a singular interval (i.e., a single point), and every odd-numbered observation corresponds to an open, bounded interval. An *interval trace* (σ, τ) consists, like a timed trace, of a trace σ and a timing τ. However, unlike for timed traces, the *positions* of the interval trace (σ, τ) are the nonnegative reals. For a position p of (σ, τ), the *observation of* (σ, τ) *at position* p is defined by $(\sigma, \tau)[p] = s_{2k}$ if $p = \sum_{i=1}^{k} t_i$, and by $(\sigma, \tau)[p] = s_{2k+1}$ if $\sum_{i=1}^{k} t_i < p < \sum_{i=1}^{k+1} t_i$. The (over)constrained temporal formulas are interpreted over interval traces without change in the definitions. Notice that in interval-based real time, the temporal operator $\bigcirc$ is best pronounced "almost": $\bigcirc\phi$ is true at position p iff for every positive real ε there is a positive real $\delta < \varepsilon$ so that ϕ is true at position $p + \delta$.

When interpreting an Alur-Dill automaton A over interval traces, the transitions of A are instantaneous, and time advances while the execution of A waits in a particular location. The exit guards are interpreted as invariants: if $(x \sim d) \in \alpha(\ell)$, then the execution of A can remain in location ℓ as long as the value of the clock x satisfies the constraint $x \sim d$. Formally, the Alur-Dill automaton A *accepts* the interval trace (σ, τ) if in the definition of timed-trace acceptance, the exit-guard and clock-progess conditions (2a) and (2b) are replaced as follows: given $i \geq 0$, for the even-numbered observations, with singular duration,

(2a) if $(x \sim d) \in \alpha(\ell_{2i})$ then $u_{2i} \sim d$, and
(2b) $v_{2i+1} = u_{2i}$;

for the odd-numbered observations, with open duration,

(2a′) if $(x \sim d) \in \alpha(\ell_{2i+1})$, then $w \sim d$ for all $u_{2i+1} < w < v_{2i+2}$, and
(2b′) $v_{2i+2} = u_{2i+1} - t_{i+1}$.

As was the case for timed traces, every Alur-Dill automaton A can be translated into a Büchi automaton $\mathcal{R}'(A)$ such that $\mathcal{R}'(A)$ accepts a trace σ iff there is a timing τ so that A accepts the interval trace (σ, τ). The interval-based region construction $\mathcal{R}'$ has the same flavor and complexity as the point-based region construction $\mathcal{R}$ [AFH96].

All results we reported for point-based real time (timed traces) apply also to interval-based real time (interval traces). In particular, the clock-tableau construction can be modified, so that the interval-based real-time model-checking problem (given a product A_S of Alur-Dill automata and a constrained temporal formula ϕ, do all interval traces that are accepted by A_S satisfy ϕ?) can be solved in polynomial space [AFH96, RS97]. In the interval-based clock tableau A'_ϕ, we use one additional clock, z, to distinguish singular from open locations: for every location ℓ, we require that $z \in \gamma(\ell)$ and either $(z = 0) \in \alpha(\ell)$ or $(z < 0) \in \alpha(\ell)$. If $(z = 0) \in \alpha(\ell)$, then location ℓ is called *singular*, because when the execution of A'_ϕ visits ℓ, it remains in ℓ only for a point in time. If $(z < 0) \in \alpha(\ell)$, then location ℓ is called *open*, because when the execution of A'_ϕ visits ℓ, it remains in ℓ for an open interval of time.

For singular locations of the interval-based clock tableau A'_ϕ, the rules that define the transition relation and guard functions are the same as for the point-based clock tableau A_ϕ. For open locations ℓ, if ℓ contains the initial copy ψ^I of the constrained until formula $\psi = (\psi_1 \mathcal{U}_{\sim c}\, \psi_2)$, then ℓ also contains ψ_1^I; in addition, either $(0, 1) \sim c$ and ℓ contains ψ_2^I, or $(x_\psi \sim c) \in \alpha(\ell)$ and each successor location of ℓ contains the stale copy ψ^S. If the open location ℓ contains the initial copy ψ^I of the earliest formula $\psi = (\rhd_I\, \psi_0)$, then $(x_\psi \in I) \in \alpha(\ell)$ and each successor of ℓ contains the stale copy ψ^S; this is the same rule as for singular ℓ, because an earliest formula cannot be fulfilled within an open interval. Stale copies are propagated and clocks are reset as in A_ϕ. Finally, if $(x_\psi = 0) \in \beta(\ell)$ for any location ℓ and formula ψ, we require that ℓ is singular.

Interleaving models. Both in integer and in real time, we adopted a *strictly monotonic* view of time: at every point in integer (real) time, every (timed/interval) trace offers a unique observation (if any). In the modeling of product systems, it is often convenient to interleave simultaneous transitions of the system components, rather than define the product of component transitions. Then, in interleaving models, several transitions of a system may occur one after the other, but all at the same point in "time." Our models are easily adopted to this *weakly monotonic* view of time, by admitting delay 0 between consecutive observations of a (timed) trace. All results we reported carry over, with only minor modifications in the algorithms. A more detailed discussion of various models for real time and their uses can be found in [AH92b].

Past temporal operators. While in integer time, past temporal operators add no expressive power, this is not the case in real time [AH92a, AFH94]. Hence, real-time logics are often defined with constrained past operators, in addition to the constrained future operators discussed in this paper. The tableau and clock-tableau constructions can naturally accommodate past operators. It follows that the addition of constrained past temporal operators does not increase model-checking complexity, neither in integer time nor in real time.

Interval-constrained until operators. An interval-constrained until operator has the form $\mathcal{U}_I$ for a *nonsingular* interval I with integer endpoints. While singular intervals cause overconstraining, nonsingular interval constraints on until operators can be model checked in exponential space. The real-time model-checking algorithm for interval-constrained until formulas, however, is complicated, and not discussed in this paper; see [AFH96].

Clock-constrained temporal logics. As an alternative to constraining temporal operators, we can add clock-reset quantifiers (also called "freeze" quantifiers) and clock constraints to temporal logic [AH94]. Since clock constraints can express overconstrained requirements of the form $\Diamond_{=c}\,\phi$, the resulting real-time logics are undecidable. However, in integer time, temporal logics with clock constraints can be model checked in exponential space, just like the overconstrained temporal formulas.

5 Conclusion

Temporal logic with clock-reset quantifiers and clock constraints has been called TPTL (integer time: exponential space; real time: undecidable) [AH94][12], with overconstrained until operators, MTL [Koy90] (integer time: exponential space; real time: undecidable) [AH93], with interval-constrained until operators, MITL (integer or real time: exponential space) [AFH96], with constrained until operators, $\text{MITL}_{0,\infty}$ (integer or real time: polynomial space) [AFH96], with the earliest operator $\rhd_I$ and its past dual $\lhd_I$, pronounced "latest," ECL [HRS98] (integer or real time: polynomial space) [RS97]. All complexities refer equally to satisfiability and model checking, and are robust with respect to point-based vs. interval-based modeling of real time, and with respect to strictly monotonic vs. weakly monotonic modeling of time.

[12] There, it is also shown that if a richer set of clock constraints is permitted —for example, constraints that compare the *sum* of two clock values with a constant— then even the integer-time model-checking problem is undecidable. This and related issues are discussed in the earlier survey [AH92b].

In integer time, all these logics are equally expressive: they define the *counter-free ω-regular* trace sets, which can also be characterized by a certain first-order monadic logic on the natural numbers [Tho90]. In real time, TPTL is strictly more expressive than MTL, which is strictly more expressive than the other three logics —MITL, MITL$_{0,\infty}$, and the future fragment of ECL. In point-based real time, MITL is strictly more expressive than both MITL$_{0,\infty}$ and ECL, which are equally expressive [Ras98]. In interval-based real time, all three logics are equally expressive [HRS98]. The interval trace sets that are definable in MITL$_{0,\infty}$/ECL —that is, definable by constrained temporal formulas— have been called *counter-free real-time ω-regular*; they have also been characterized by a certain first-order monadic logic on the reals [HRS98].

The observation that constrained until formulas (MITL$_{0,\infty}$) and earliest formulas (ECL) are interdefinable means that we could have omitted one or the other from our discussion. We included both, because each offers the direct specification of an important class of timing requirements, and because each offers independent insights into the clock-tableau construction. The key ideas behind the translations are given by the following equivalences: the earliest formula $\rhd_{[1,2]} \phi$ is equivalent to the conjunction of the two constrained until formulas $\Diamond_{\leq 2} \phi$ and $\Box_{<1} \neg\phi$; the constrained until formula $\phi_1 \mathcal{U}_{\leq 1} \phi_2$ is equivalent to the conjunction of the unconstrained until formula $\phi_1 \mathcal{U} \phi_2$ and the earliest formula $\rhd_{\leq 1} \phi_2$; the constrained until formula $\phi_1 \mathcal{U}_{\geq 1} \phi_2$ is equivalent to the conjunction of $\Box_{<1} \phi_1$ and $\Box_{\leq 1} (\phi_2 \vee (\phi_1 \wedge (\phi_1 \mathcal{U} \phi_2)))$.

Acknowledgments. The author thanks Jean-Francois Raskin for several valuable suggestions on a draft of this paper, Prasad Sistla for repeatedly pointing out the importance of polynomial-space fragments of real-time logics, and Rajeev Alur for uncountable hours of research and discussion on the topic of real-time verification.

References

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

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 1–13. Springer-Verlag, 1994.

[AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AH92a] R. Alur and T.A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 177–186. IEEE Computer Society Press, 1992.

[AH92b] 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. Springer-Verlag, 1992.

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

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

[BKP86] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the 13th Annual Symposium on Principles of Programming Languages*, pages 173–183. ACM Press, 1986.

[Büc62] J.R. Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Proceedings of the First International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–11. Stanford University Press, 1962.

[CK96] E.M. Clarke and R.P. Kurshan. Computer-aided verification. *IEEE Spectrum*, 33(6):61–67, 1996.

[Eme90] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[Eme92] E.A. Emerson. Real time and the μ-calculus. 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 176–194. Springer-Verlag, 1992.

[EMSS90] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 136–145. Springer-Verlag, 1990.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the Seventh Annual Symposium on Principles of Programming Languages*, pages 163–173. ACM Press, 1980.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[HRS98] T.A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *ICALP 97: Automata, Languages, and Programming*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite-state concurrent programs satisfy their linear specification. In *Proceedings of the 12th Annual Symposium on Principles of Programming Languages*, pages 97–107. ACM Press, 1985.

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

[Ras98] J.-F. Raskin. Personal communication, 1998.

[RS97] J.-F. Raskin and P.-Y. Schobbens. State-clock logic: a decidable real-time logic. In O. Maler, editor, *HART 97: Hybrid and Real-time Systems*, Lecture Notes in Computer Science 1201, pages 33–47. Springer-Verlag, 1997.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 319–327. IEEE Computer Society Press, 1988.

[Sav70] W.J. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–194, 1970.

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

[Sis83] A.P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems.* PhD thesis, Harvard University, 1983.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier Science Publishers, 1990.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 694–715. Springer-Verlag, 1994.

[Wol82] P. Wolper. *Synthesis of Communicating Processes from Temporal-Logic Specifications.* PhD thesis, Stanford University, 1982.