The Expressive Power of Clocks*

Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi

Computer Science Department, Cornell University, Ithaca, NY 14853

Abstract. We investigate the expressive power of timing restrictions on labeled transition systems. In particular, we show how constraints on clock variables together with a uniform liveness condition — the divergence of time — can express Büchi, Muller, Streett, Rabin, and weak and strong fairness conditions on a given labeled transition system. We then consider the effect, on both timed and time-abstract expressiveness, of varying the following parameters: time domain (discrete or dense), number of clocks, number of states, and size of constants used in timing restrictions.

1 Introduction

We study the expressive power of labeled transition systems with clocks, so-called *timed safety automata* [11]. Timed safety automata are timed automata [2] without acceptance conditions; their liveness is imposed uniformly as a progress condition on time. Timed safety automata have been used extensively for the specification and verification of real-time systems [6, 7, 11, 9]. It has been argued that with the explicit consideration of time, acceptance conditions are no longer useful abstractions to enforce liveness [8], and this paper corroborates that belief.

We look at both the time-abstract expressive power (Section 3) and the timed expressive power (Section 4) of timed safety automata. In the time-abstract view, a timed safety automaton defines a set of infinite words over the input alphabet A; in the timed view, each input symbol is labeled with a time stamp, and a timed safety automaton defines a set of infinite words over the alphabet $A \times \mathbb{T}$, where $\mathbb{T}$ is the time domain. We look at several orthogonal parameters that affect the expressiveness of timed safety automata: number of automaton states, number of clocks, size of time constants that occur in clock constraints, and time domain (discrete or dense). For instance, we examine the hierarchy obtained by fixing the number of states, and letting the number of clocks vary, and the hierarchy obtained by considering only the number of clocks.

* This research was supported in part by the National Science Foundation under grant CCR-9200794, by the United States Air Force Office of Scientific Research under contract F49620-93-1-0056, by the Defense Advanced Research Projects Agency under grant NAG2-892, and by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, Contract Number DAAL03-91-C-0027.

The full version of this paper is available from the Department of Computer Science, Cornell University, Ithaca, NY 14853, as Technical Report TR95-1496.

Time-abstract expressiveness. Liveness conditions are typically imposed on state-transition structures as conditions on the set of states that are visited infinitely often. The clock view, by contrast, ensures the liveness of a state-transition structure through the divergence of time under clock constraints. First, we give direct translations of generalized Büchi, Muller, Streett, and Rabin acceptance conditions, and of weak and strong fairness conditions, into clock constraints. We show that these liveness conditions can be enforced on a given state-transition structure using either just two clocks, or time constants of size at most 2. Second, we show that a single clock and the time constant 1 suffice to express all ω-regular languages. Indeed, both of these results hold for discrete and for dense time.

Over dense time, the expressive power of clocks is even greater. We show that dense-time clocks can be used to enforce any ω-regular liveness condition on a given state-transition structure. This implies that, in dense time, every ω-regular language is accepted by a timed safety automaton with a single state. The same is not true for discrete time, where, surprisingly, for any fixed state transition structure, two clocks are as expressive as any number of clocks.

Timed expressiveness. The timed language of a timed safety automaton is limit-closed [11], and therefore the timed expressiveness of timed safety automata is strictly less than that of timed automata [2]. In dense time, for any natural number k, there are timed languages that are accepted by a timed safety automaton with $k+1$ clocks, but not by any timed safety automaton with k clocks. We thus obtain an infinite hierarchy of dense-time languages. In discrete time, the timed languages that are accepted by timed safety automata are precisely the limit-closed *timed ω-regular languages* [4, 5]. In contrast with dense time, however, a single discrete-time clock suffices to accept any limit-closed timed ω-regular language. On the other hand, for any fixed number of states, increasing the number of clocks does increase expressive power.

Two-way timed automata are studied in [3]. An infinite hierarchy of timed expressiveness is obtained, based upon the number of alternations. Clock hierarchies of a different nature are studied in [1]. Various types of timed and time-abstract observational equivalence are considered for observers with a limited number of clocks.

2 Labeled Transition Systems

A *labeled transition system* $\mathcal{A}$ is a quadruple $(S, A, \rightarrow, S_0)$, for a set S of *states*, a finite *alphabet* A of *events*, a *transition relation* $\rightarrow \subset S \times A \times S$, and a set $S_0 \subset S$ of *initial states*. We write $s \xrightarrow{a} t$ in place of $(s, a, t) \in \rightarrow$. A *run* r of $\mathcal{A}$ is an infinite sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$ of states $s_i \in S$ and events $a_i \in A$ such that $s_0 \in S_0$. The run r *generates* the word $a_0 a_1 a_2 \ldots$, denoted $\overline{a}$. The *language* $[\![\mathcal{A}]\!] \subset A^\omega$ of the labeled transition system $\mathcal{A}$ is the set of infinite words that are generated by runs of $\mathcal{A}$.

Acceptance conditions. The labeled transition system $\mathcal{A}$ is a *finite automaton* if the state space S is finite. The liveness of finite automata is commonly enforced by acceptance conditions. The *infinite visitation set* $\mathsf{inf}(r)$ of the run r is the set of states that occur infinitely often along r. A *generalized Büchi condition* $\mathcal{B}$ for the labeled transition system $\mathcal{A}$ is a set of subsets of S (the Büchi acceptance condition $\mathcal{B}$ is *standard* if $|\mathcal{B}| = 1$). The run r is *$\mathcal{B}$-accepting* if for every $I \in \mathcal{B}$, $\mathsf{inf}(r) \cap I \neq \emptyset$. A *Muller condition* $\mathcal{M}$ is also a set of subsets of S. The run r is *$\mathcal{M}$-accepting* if $\mathsf{inf}(r) \in \mathcal{M}$. A *Rabin condition* $\mathcal{R}$ is a set of pairs of subsets of S. The run r is *$\mathcal{R}$-accepting* if there exists a pair $(I, J) \in \mathcal{R}$ such that $\mathsf{inf}(r) \cap I \neq \emptyset$ and $\mathsf{inf}(r) \cap J = \emptyset$. A *Streett condition* $\mathcal{S}$ is also a set of pairs of subsets of S, representing the negation of the corresponding Rabin condition: the run r is *$\mathcal{S}$-accepting* if for all $(I, J) \in \mathcal{S}$, if $\mathsf{inf}(r) \cap I \neq \emptyset$ then $\mathsf{inf}(r) \cap J \neq \emptyset$. The *Büchi* (*Muller*; *Rabin*; *Streett*) *language* $[\![\mathcal{A}, \mathcal{X}]\!] \subset A^\omega$ of the labeled transition system $\mathcal{A}$ under the acceptance condition $\mathcal{X} = \mathcal{B}$ ($\mathcal{M}$; $\mathcal{R}$; $\mathcal{S}$) is the set of infinite words that are generated by $\mathcal{X}$-accepting runs of $\mathcal{A}$. The Büchi (Muller; Rabin; Streett) languages of finite automata are the ω-regular languages [12].

Fairness conditions. The labeled transition system $\mathcal{A}$ is *event-recording* if for each state $s \in S$ there is a label $a_s \in A$ such that $t \xrightarrow{a} s$ implies $a = a_s$. For every labeled transition system $\mathcal{A} = (S, A, \rightarrow, S_0)$, there is an event-recording transition system $\mathcal{A}' = (S \times A, A, \Rightarrow, S_0 \times A)$, with $(s, a) \overset{b}{\Rightarrow} (t, b)$ iff $s \xrightarrow{b} t$, such that $\mathcal{A}$ and $\mathcal{A}'$ define the same language. The liveness of event-recording transition systems is commonly enforced by fairness conditions. The *enabling set* $\mathsf{enabled}(a)$ of the event a is the set of states s such that there is a successor state t with $s \xrightarrow{a} t$; the *completion set* $\mathsf{taken}(a)$ is the set of states s such that there is a predecessor state t with $t \xrightarrow{a} s$. A *weak-fairness condition* $\mathcal{F}_W$ for the labeled transition system $\mathcal{A}$ is a set of events. A run r is *weakly $\mathcal{F}_W$-fair* if for each $a \in \mathcal{F}_W$, infinitely many of the states of r lie in $\neg\mathsf{enabled}(a) \cup \mathsf{taken}(a)$. If $\mathcal{A}$ is a finite automaton, this corresponds to acceptance by the generalized Büchi condition that contains, for each event $a \in \mathcal{F}_W$, the set $\neg\mathsf{enabled}(a) \cup \mathsf{taken}(a)$. A *strong-fairness condition* $\mathcal{F}_S$ is also a set of events. A run r is *strongly $\mathcal{F}_S$-fair* if for every $a \in \mathcal{F}_S$, if infinitely many of the states of r lie in $\mathsf{enabled}(a)$, then infinitely many lie in $\mathsf{taken}(a)$. If $\mathcal{A}$ is a finite automaton, this corresponds to acceptance by the Streett condition $\mathcal{S}$ that contains, for each event $a \in \mathcal{F}_S$, the pair $(\mathsf{enabled}(a), \mathsf{taken}(a))$. The *weakly* (*strongly*) *fair language* $[\![\mathcal{A}, \mathcal{F}]\!]$ of the labeled transition system $\mathcal{A}$ under the fairness condition $\mathcal{F} = \mathcal{F}_W$ ($\mathcal{F}_S$) is the set of infinite words that are generated by weakly (strongly) $\mathcal{F}$-fair runs of $\mathcal{A}$.

Timing conditions. We consider both the discrete time domain $\mathbb{T} = \mathbb{N}$ and the dense time domain $\mathbb{T} = \mathbb{R}_{\geq 0}$. Let C be a set of $\mathbb{T}$-valued variables called *clocks*. The *clock constraints* are the formulas defined by the grammar

$$\varphi ::= x \leq n \mid x \geq n \mid x \leq y + n \mid \neg\varphi \mid \varphi \wedge \varphi$$

where x and y are clocks, and n is an integer. A *closed clock constraint* is a conjunction of formulas of the form $x \leq n$ and $x \geq n$. A *clock valuation* $\nu : C \rightarrow \mathbb{T}$ is

a map that assigns to each clock x a time value $\nu(x)$. The clock valuation ν, then, assigns to each clock constraint φ a truth value $\nu(\varphi)$. Given a clock valuation ν and a time delay $\delta \in \mathbb{T}$, the clock valuation $\nu+\delta$ assigns to each clock x the time value $(\nu+\delta)(x) = \nu(x)+\delta$. Given a set $X \subset C$ of clocks, the clock valuation $\nu[X := 0]$ assigns to each clock $x \in X$ the value 0, and to each clock $x \notin X$ the time value $\nu(x)$. The set of all clock valuations is denoted $\mathcal{V}_C$. It is naturally isomorphic to $\mathbb{T}^{|C|}$. Closed clock constraints specify closed subsets of $\mathcal{V}_C$. An *atomic clock command* ϕ is a pair (φ, X), where the *transition guard* φ is a clock constraint and the *reset set* X is a subset of C. The atomic clock command ϕ defines a partial function on the set $\mathcal{V}_C$ of clock valuations: if $\nu(\varphi) = \mathit{true}$ then $\phi(\nu) = \nu[X := 0]$; otherwise $\phi(\nu)$ is undefined. A *clock command* ψ is a finite set of atomic clock commands. The clock command ψ defines a relation on the set $\mathcal{V}_C$ of clock valuations: $(\mu, \nu) \in \psi$ iff ψ contains an atomic clock command ϕ such that $\nu = \phi(\mu)$.

A *timing condition* $\mathcal{T} = (C, \psi)$ for the labeled transition system $\mathcal{A}$ consists of a finite set C of clocks and a function ψ that assigns to each transition $s \xrightarrow{a} t$ of $\mathcal{A}$ a clock command $\psi(s, a, t)$. The timing condition $\mathcal{T}$ is *closed* if every clock constraint of every clock command $\psi(s, a, t)$ is closed. If $\mathcal{A}$ is a finite automaton, then the pair $(\mathcal{A}, \mathcal{T})$ is called a *timed safety automaton* [11].

A $\mathbb{T}$-*timing* is an infinite sequence $\overline{\delta} \in \mathbb{T}^\omega$ of time delays such that the sum $\Sigma_{i \geq 0} \delta_i$ diverges. A $\mathbb{T}$-*timed word* is a pair $(\overline{a}, \overline{\delta})$ where $\overline{a} \in A^\omega$ is an infinite word and $\overline{\delta}$ is a timing. A $\mathbb{T}$-timed word $(\overline{a}, \overline{\delta})$ is *accepted* by $(\mathcal{A}, \mathcal{T})$ if there is a run $r = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ of $\mathcal{A}$ and a sequence $\overline{\nu}$ of clock valuations such that ν_0 is the constant mapping $\lambda x.0$, and for all i, $(\nu_i + \delta_i, \nu_{i+1}) \in \psi(s_i, a_i, s_{i+1})$. In this case, the pair $(r, \overline{\nu})$ is called a *divergent execution* of $(\mathcal{A}, \mathcal{T})$. The *timed language* $[\![\mathcal{A}, \mathcal{T}]\!]^t_{\mathbb{T}}$ of $(\mathcal{A}, \mathcal{T})$ under time domain $\mathbb{T}$ is the set of $\mathbb{T}$-timed words accepted by $(\mathcal{A}, \mathcal{T})$. The *untimed language* $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{T}}$ is the set of words $\overline{a} \in A^\omega$ such that for some $\mathbb{T}$-timing $\overline{\delta}$, $(\overline{a}, \overline{\delta}) \in [\![\mathcal{A}, \mathcal{T}]\!]^t_{\mathbb{T}}$.

Some previous expressiveness results. First, if the timing condition $\mathcal{T}$ is *closed*, then the choice of time domain is irrelevant, because then $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}} = [\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{R}_{\geq 0}}$ [10]. Second, in the discrete time domain, the set of *timed* languages accepted by timed safety automata is the class of limit-closed timed ω-regular languages [4, 5]. Third, in either time domain, the class of *untimed* languages accepted by timed safety automata is the class of ω-regular languages. This is because every timed safety automaton $\mathcal{A} = (S, A, \rightarrow, S_0)$ has a finite bisimulation [2].

Let h be the largest constant appearing in the clock constraints of $\mathcal{T}$. Define an equivalence relation $\equiv_h$ on the set $\mathcal{V}_C$ of clock valuations by $\nu \equiv_h \mu$ iff for every clock constraint φ containing no constant greater than h, $\nu(\varphi) = \mu(\varphi)$. Roughly speaking, two valuations are equivalent iff they agree on the ordering of the fractional parts of the clocks, they agree on which clocks are integers, and they agree on the the integer part of each clock with value no more than h. An equivalence class of $\equiv_h$ is called a *region*. The region of ν is denoted $\overline{\nu}$. The *region automaton* $\mathit{Reg}(\mathcal{A}, \mathcal{T})_{\mathbb{T}}$ for $(\mathcal{A}, \mathcal{T})$ is the finite automa-

ton $(S \times (\mathcal{V}_C/\equiv_h), A, \rightarrow', S_0 \times \{\overline{\lambda x.0}\})$. The transition relation is defined by $((s,\overline{\mu}),a,(t,\overline{\nu})) \in \rightarrow'$ iff $s \xrightarrow{a} t$ and there exist valuations $\xi \in \overline{\mu}$ and $\zeta \in \overline{\nu}$, a duration $\delta \in \mathbb{T}$, and an atomic clock command $(\varphi, X) \in \psi(s,a,t)$ such that $(\xi+\delta)(\varphi) = \mathit{true}$ and $\zeta = (\xi+\delta)[X := 0]$. Let $\mathcal{B}$ be the generalized Büchi condition that requires each clock x either to be infinitely often greater than h, or both infinitely often 0 and infinitely often nonzero. The Büchi condition $\mathcal{B}$ in effect enforces time divergence on $(\mathcal{A},\mathcal{T})$. Every divergent execution of $(\mathcal{A},\mathcal{T})$ corresponds naturally to a run of $\mathit{Reg}(\mathcal{A},\mathcal{T})_{\mathbb{T}}$ satisfying $\mathcal{B}$, and vice versa. The region automaton is the main tool for the analysis of timed safety automata [2].

3 Untimed Languages

We study the expressiveness of timing conditions with respect to untimed languages over the alphabet A.

3.1 Time Enough for Liveness

We give several ways to replace acceptance conditions on finite automata, or fairness conditions on labeled transition systems, by timing conditions. First, we decorate an existing state transition structure with timing constraints to impose a liveness condition, using as few clocks as possible. Second, we accomplish the same task using clock constraints with constants as small as possible. Third, we show that a single clock and the constant 1 suffice if we can change the state-transition structure. In this subsection, all results hold for both the discrete time domain $\mathbb{T} = \mathbb{N}$ and the dense time domain $\mathbb{T} = \mathbb{R}_{\geq 0}$.

Minimizing the number of clocks.

Theorem 3.1 *For every finite automaton $\mathcal{A}$ with state space S, and every generalized Büchi (Muller; Rabin; Streett) acceptance condition $\mathcal{X}$ for $\mathcal{A}$, there is a closed timing condition $\mathcal{T}$, using exactly two clocks and no constants greater than $|\mathcal{X}|$ ($|S||\mathcal{X}|$; $|\mathcal{X}|$; $2^{|\mathcal{X}|}|\mathcal{X}|$) in clock constraints, such that $[\![\mathcal{A},\mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A},\mathcal{X}]\!]$.*

Proof. Two clocks x and y can be used to store a positive integer value m by use of the invariant $|x-y| = m$. After each transition, we force one of the clocks to have the value m, and one to have value 0. To change the value from m to n we use the atomic clock commands

$$((x = m+n) \wedge (y = n), \{x\}) \text{ and } ((y = m+n) \wedge (x = n), \{y\}).$$

Notice that these commands allow at least one unit of time to pass.

We implement an acceptance condition by decomposing it into an infinite sequence of "helpful transitions." Two clocks are used to store a value that corresponds to the next required helpful transition. When this transition is taken, the value is changed to the following helpful transition. In this way, time diverges iff infinitely many helpful transitions are taken iff the acceptance condition is

fulfilled. We give the proof for Streett acceptance, leaving the others to the full paper.

Let $\mathcal{X}$ be a Streett acceptance condition. A run $r = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ of $\mathcal{A}$ is accepted by $\mathcal{X}$ iff there is a subset $\mathcal{X}_{fin}$ of $\mathcal{X}$ such that

- for each $(I, J) \in \mathcal{X}_{fin}$, $\mathsf{inf}(r) \cap I = \emptyset$, and
- for each $(I, J) \in \mathcal{X} \setminus \mathcal{X}_{fin}$, $\mathsf{inf}(r) \cap J \neq \emptyset$.

Enumerate the possible values for $\mathcal{X}_{fin}$ (i.e., the subsets of $\mathcal{X}$), as $\mathcal{X}_1, \ldots, \mathcal{X}_n$. For each $\mathcal{X}_i$, enumerate the J such that $(I, J) \in \mathcal{X} \setminus \mathcal{X}_i$ for some I, as $J_{i,1}, \ldots, J_{i,m(i)}$. For $\mathcal{X}_{fin} = \mathcal{X}_i$, the helpful transitions are the transitions into the $J_{i,j}$. If entry to $J_{i,j}$ was the previous helpful transition, then the next helpful transition is defined to be entry into $J_{i,j+1 \bmod m(i)}$, where we take $\{1, \ldots, m(i)\}$ to be the range of mod $m(i)$. We store the two values i and j, corresponding respectively to the choice of $\mathcal{X}_{fin}$ and the next J to be visited.

Let $\gamma : \{(i,j) | 1 \leq i \leq n \wedge 1 \leq j \leq m(i)\} \to \{1, \ldots, n|X|\}$ be any one-to-one map. For each i, let $Bad_i = \bigcup\{I \mid \exists J.(I, J) \in \mathcal{X}_i\}$. To each transition $s \xrightarrow{a} t$ we assign four classes of atomic clock commands. First, to disallow infinitely many visits to Bad_i, we have

$$((x = 0) \wedge (y = 0), \emptyset).$$

Each run begins with some finite number of transitions of this type—a run with infinitely many converges.

Second, to guess $\mathcal{X}_i$ we have, for each i with $1 \leq i \leq n$, the atomic clock command

$$((x = \gamma(i,1)) \wedge (y = \gamma(i,1)), \{y\}).$$

Third, once $\mathcal{X}_i$ is chosen, we must not allow further visits to Bad_i. We must also not let time advance until the next $J_{i,j}$ is reached. For each i, j with $t \notin Bad_i$ and $t \notin J_{i,j}$, we have the atomic clock commands

$$((x = \gamma(i,j)) \wedge (y = 0), \emptyset) \text{ and } ((y = \gamma(i,j)) \wedge (x = 0), \emptyset).$$

Finally, to advance time when the next $J_{i,j}$ is met, we have for each i, j with $t \notin Bad_i$ and $t \in J_{i,j}$,

$$((x = \gamma(i,j) + \gamma(i, j+1 \bmod m(i))) \wedge (y = \gamma(i, j+1 \bmod m(i))), \{x\}),$$

$$((y = \gamma(i,j) + \gamma(i, j+1 \bmod m(i))) \wedge (x = \gamma(i, j+1 \bmod m(i))), \{y\}).$$

This completes the construction. Notice that if we did not allow time steps of duration zero, the proof would need only minor modifications. □

Corollary 3.2 *For every event-recording labeled transition system $\mathcal{A}$, and every weak or strong-fairness condition $\mathcal{F}$ for $\mathcal{A}$, there is a closed timing condition $\mathcal{T}$ using exactly two clocks such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A}, \mathcal{F}]\!]$.*

Minimizing the size of constants used in clock constraints. While only two clocks are used in Theorem 3.1, the constants used in clock constraints are exponential in the number of components of the acceptance condition. If we wish to bound the constants by 2, we can get by with a logarithmic number of clocks (linear in the case of Streett acceptance).

Theorem 3.3 *Let $\mathcal{A}$ be a finite automaton with state space S.*

1. *For every generalized Büchi (Muller; Rabin) acceptance condition $\mathcal{X}$ for $\mathcal{A}$, there is a closed timing condition $\mathcal{T}$ using $1 + \lceil \log_2 |\mathcal{X}| \rceil$ $(1 + \lceil \log_2 |\mathcal{X}||S| \rceil$; $1 + \lceil \log_2 |\mathcal{X}| \rceil)$ clocks, and no constants greater than 2 in clock constraints, such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathrm{T}} = [\![\mathcal{A}, \mathcal{X}]\!]$.*
2. *For every Streett acceptance condition $\mathcal{S}$ for $\mathcal{A}$, there is a closed timing condition $\mathcal{T}$ using $|\mathcal{S}|$ clocks, and no constants greater than 1 in clock constraints, such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathrm{T}} = [\![\mathcal{A}, \mathcal{S}]\!]$.*

Proof sketch. A value in $\{1, ..., n\}$ may be stored in $1 + \lceil \log_2 n \rceil$ clocks by using $\lceil \log_2 n \rceil$ clocks for its binary encoding, and one additional clock to mediate increments. So for part 1, the bounds on the number of clocks result from the constants required in Theorem 3.1. For 2, use one clock $x_{(I,J)}$ for each element (I, J) of $\mathcal{S}$. Allow entry to I only if $x_{(I,J)} \leq 1$, and reset $x_{(I,J)}$ only upon entry to J. □

Corollary 3.4 *For every event-recording labeled transition system $\mathcal{A}$, and every weak or strong-fairness condition $\mathcal{F}$ for $\mathcal{A}$, there is a closed timing condition $\mathcal{T}$ using $|\mathcal{F}|$ clocks, and no constants greater than 1 in clock constraints, such that $[\![\mathcal{A}, \mathcal{F}]\!] = [\![\mathcal{A}, \mathcal{T}]\!]_{\mathrm{T}}$.*

Changing the state transition structure. The previous constructions kept intact the original state transition structure. The cost of this was in the number of clocks or the size of the constants used in clock constraints. If we alter the state transition structure, then we need only one clock and the constant 1 in clock constraints. Since every ω-regular language is accepted by a finite automaton with a Streett acceptance condition with one pair, we have the following corollary to the second part of Theorem 3.3.

Corollary 3.5 *For every finite automaton $\mathcal{A}$, and every generalized Büchi, Muller, Rabin, or Streett acceptance condition $\mathcal{X}$ for $\mathcal{A}$, there is a timed safety automaton $(\mathcal{C}, \mathcal{T})$, with $\mathcal{T}$ closed, using one clock, and no constants greater than 1 in clock constraints, such that $[\![\mathcal{C}, \mathcal{T}]\!]_{\mathrm{T}} = [\![\mathcal{A}, \mathcal{X}]\!]$.*

By replicating the state space, a generalized Büchi condition may be replaced by a standard one. With this and the proof of the second part of Theorem 3.3, we can implement any fairness condition with one clock and small constants.

Theorem 3.6 *For every event-recording labeled transition system $\mathcal{A}$, and every weak or strong-fairness condition $\mathcal{F}$ for $\mathcal{A}$, there is a labeled transition system $\mathcal{C}$ and a closed timing condition $\mathcal{T}$ for $\mathcal{C}$ using one clock, and no constants greater than 1 in clock constraints, such that $[\![\mathcal{C}, \mathcal{T}]\!]_{\mathrm{T}} = [\![\mathcal{A}, \mathcal{F}]\!]$.*

3.2 Dense-Time Hierarchies

The power of dense time. Previously, we replaced liveness conditions with timing conditions. When the dense time domain $\mathbb{T} = \mathbb{R}_{\geq 0}$ is used, clocks acquire greater power. They may be used to enforce any ω-regular liveness condition on a given finite automaton.

Theorem 3.7 *For every finite automaton $\mathcal{A}$, and every ω-regular language L, there is a timing condition $\mathcal{T}$ such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{R}_{\geq 0}} = [\![\mathcal{A}]\!] \cap L$.*

Proof sketch. Let $\mathcal{A}' = (S', A, \rightarrow', S_0)$ be a finite automaton and $\mathcal{B}'$ a standard Büchi acceptance condition for $\mathcal{A}'$ such that $[\![\mathcal{A}', \mathcal{B}']\!] = L$. While performing a computation on $\mathcal{A}$, we simulate the computation of $\mathcal{A}'$ by using one clock x_s for each state s of $\mathcal{A}'$. The clock with the largest value corresponds to the current state of $\mathcal{A}'$. The current state of $\mathcal{A}'$ can be determined by the clock constraints $In_{s'}$, defined for each $s' \in S'$ by $In_{s'} = \bigwedge_{t' \in S' \setminus \{s'\}} (x_{s'} > x_{t'} \wedge x_{t'} > 0)$. When a transition is taken, all clocks but the one corresponding to the target state are reset. We use one additional clock $x_{\mathcal{B}'}$ to enforce the acceptance condition of $\mathcal{A}'$. No transition is enabled if $x_{\mathcal{B}'}$ is greater than one, and $x_{\mathcal{B}'}$ is reset only when control enters $\mathcal{B}'$. The density of the time domain allows an arbitrary number of transitions in between entries into $\mathcal{B}'$. □

By applying Theorem 3.7 to the one-state automaton with language A^ω, we obtain the following counterpart to Corollary 3.5.

Corollary 3.8 *For every ω-regular language L, there is a timed safety automaton $(\mathcal{A}, \mathcal{T})$ with one state such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{R}_{\geq 0}} = L$.*

The state hierarchy for a fixed number of clocks. The following theorem shows that increasing the number of states, while holding the number of clocks fixed, yields an increase in expressive power.

Theorem 3.9 *For every $k, n \in \mathbb{N}$, the class of languages accepted by timed safety automata with k states and n clocks over the dense time domain $\mathbb{R}_{\geq 0}$ is properly contained in the class accepted with $k+1$ states and n clocks.*

The clock hierarchy for a fixed number of states.

Theorem 3.10 *For every $k, n \in \mathbb{N}$, the class of languages accepted by timed safety automata with k states and n clocks over the dense time domain $\mathbb{R}_{\geq 0}$ is properly contained in the class accepted with k states and $n+1$ clocks.*

3.3 Discrete-Time Hierarchies

The power of discrete time. Suppose $\mathcal{A}$ is a finite automaton with state space S, and $\sim$ is an equivalence relation on S. The *infinitely often change class* acceptance condition induced by $\sim$ on $\mathcal{A}$ is the generalized Büchi acceptance

condition $\mathcal{B}_\sim = \{S \setminus \{t \in S \mid t \sim s\} \mid s \in S\}$. The *multiplicity* of $\sim$ is the number of elements in the largest equivalence class of $\sim$. A *k-iocc* language is a language accepted by a finite automaton with the iocc acceptance condition induced by an equivalence relation of multiplicity k. The language $(a_1 \cdots a_{k+1})^* a_1^\omega$ is $k+1$-iocc but not k-iocc. So the k-iocc languages form a strict hierarchy whose union over all k is the class of ω-regular languages. The k-iocc languages are incomparable with the class of languages defined by deterministic Büchi automata, and as far as we know, have not been studied previously.

Theorem 3.11 *For every $k \geq 1$, the set of languages defined by timed safety automata with k states over the discrete time domain $\mathbb{N}$ coincides with the set of k-iocc languages.*

Proof sketch. In discrete time, divergence is equivalent to infinitely many nonzero time steps, which in turn is almost equivalent to infinitely many changes of region. Modifying the region automaton by duplicating regions that are their own successors under time steps, divergence may be expressed by a k-iocc acceptance condition. To show that every k-iocc language $L = [\![\mathcal{A}, \mathcal{B}_\sim]\!]$ is the language of a k-state timed safety automaton $(\mathcal{C}, \mathcal{T})$ over discrete time, use two clocks to record the $\sim$-equivalence class, as in the proof of Theorem 3.1, and use the k states of $\mathcal{C}$ to record the particular member of the equivalence class. □

The state hierarchy for a variable number of clocks.

Corollary 3.12 *For every $k \in \mathbb{N}$, there is a language accepted by a timed safety automaton with $k+1$ states over the discrete time domain $\mathbb{N}$ which is not accepted by any timed safety automaton with k states over $\mathbb{N}$.*

Hence no finite number of states suffices to define all ω-regular languages by timed safety automata over the discrete time domain $\mathbb{N}$. This is in marked contrast to the dense-time case (see Corollary 3.8), where one state suffices.

The clock hierarchy for a variable number of states collapses. While increasing the number of dense-time clocks increases the expressive power of timed safety automata, increasing the number of discrete-time clocks does not. This is because two clocks may be used to encode the region component of a discrete-time region automaton.

Theorem 3.13 *For every timed safety automaton $(\mathcal{A}, \mathcal{T})$, there is a timing condition $\mathcal{T}'$ using exactly two clocks such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}} = [\![\mathcal{A}, \mathcal{T}']\!]_{\mathbb{N}}$.*

4 Timed Languages

We study the expressiveness of timed automata with respect to timed languages. We provide proofs of two folk theorems: namely that increasing the number

of clocks increases expressiveness in the dense time case, but not for discrete time. We also show that for both time domains increasing either the number of states with a fixed number of clocks, or vice versa, yields a strict hierarchy of expressiveness.

The dense-clock hierarchy for a variable number of states. Additional clocks in a dense time domain add expressiveness that cannot be captured by increasing the state space, since events can occur arbitrarily close together.

Theorem 4.1 *For every $n \in \mathbb{N}$, the class of timed languages accepted by timed safety automata with n clocks over the dense time domain $\mathbb{R}_{\geq 0}$ is properly contained in the class accepted by automata with $n+1$ clocks.*

Proof sketch. Intuitively, a timed safety automaton with n clocks cannot enforce correct timing constraints over $n+1$ concurrent events. Let $\mathcal{L}_{n+1}$ be the timed language consisting of all timed sequences $(\overline{a}, \overline{\delta})$ such that for each $i \geq 0$ there are exactly $n+1$ a events occurring over the time interval $[i, i+1)$, and each a event is followed by another a event exactly 1 time unit later. It is easy to construct an automaton with $n+1$ clocks accepting this language.

On the other hand, consider any automaton for $\mathcal{L}_{n+1}$. It accepts the timed word where the i-th a occurs at time $1/i$ for $1 \leq i \leq n+1$. The timing conditions on each taken transition must imply $x = m$ for some clock x and integer m. Therefore at each time $1 + 1/i$ for $1 \leq i \leq n+1$ there is a clock with integer value. Therefore there must be at least $n+1$ clocks. □

The discrete-clock hierarchy for a variable number of states collapses.

Theorem 4.2 *Every timed language accepted by a timed automaton over the discrete time domain $\mathbb{N}$ is also accepted by an automaton interpreted over $\mathbb{N}$ with only one clock.*

Proof sketch. In discrete time, clock values can all be encoded in the states of an automaton, because one region may follow another only by a specific set of integral time steps. Thus, given a timed safety automaton $(\mathcal{A}, \mathcal{T})$, we may construct a timing condition $\mathcal{T}'$ over one clock x such that $[\![Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}, \mathcal{T}']\!]^t_{\mathbb{N}} = [\![\mathcal{A}, \mathcal{T}]\!]^t_{\mathbb{N}}$, by using x to allow or disallow transitions between regions. □

The state hierarchy for a variable number of clocks. Consideration of the timed language $(a, 0)^k (b, 1)^\omega$ gives the following theorem.

Theorem 4.3 *For every $k \in \mathbb{N}$, and either time domain $\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}_{\geq 0}$, the class of timed languages accepted by timed safety automata with k states over time domain $\mathbb{T}$ is properly contained in the class accepted with $k+1$ states.*

Theorem 4.3 fails if we change the model so that time steps of duration 0 are not allowed. In this case, every timed language accepted by some timed safety automaton is accepted by one with only one state. The proof mimics that of Theorem 3.7. This is the only instance known to the authors in which the strictness of the progression of time has a bearing upon the theory.

The state hierarchy for a fixed number of clocks. We now show that for both time domains, increasing the number of states for a fixed number of clocks strictly increases expressiveness.

Theorem 4.4 *For every $k, n \in \mathbb{N}$, and either time domain $\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}_{\geq 0}$, the class of timed languages accepted by timed safety automata with k states and n clocks is properly contained in the class accepted by automata with $k+1$ states and n clocks.*

The clock hierarchy for a fixed number of states.

Theorem 4.5 *For every $k, n \in \mathbb{N}$, and either time domain $\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}_{\geq 0}$, the class of timed languages accepted by timed safety automata with k states and n clocks is properly contained in the class accepted by automata with k states and $n+1$ clocks.*

5 Summary

The first table summarizes the results of Section 3.1, regarding the replacement of acceptance conditions by timing conditions. Here S refers to the state space.

Acceptance condition $\mathcal{X}$	Number of Clocks	Constants
Büchi	2	$\lvert\mathcal{X}\rvert$
	$\log \lvert\mathcal{X}\rvert$	2
Muller	2	$\lvert\mathcal{X}\rvert\lvert S\rvert$
	$\log \lvert\mathcal{X}\rvert\lvert S\rvert$	2
Rabin	2	$\lvert\mathcal{X}\rvert$
	$\log \lvert\mathcal{X}\rvert$	2
Streett	2	$2^{\lvert\mathcal{X}\rvert}\lvert\mathcal{X}\rvert$
	$\lvert\mathcal{X}\rvert$	1

For $k, n \in \mathbb{N}$, and $\mathbb{T} \in \{\mathbb{N}, \mathbb{R}_{\geq 0}\}$, let $\mathcal{U}(k, n, \mathbb{T})$ be the set of untimed languages, and let $\mathcal{T}(k, n, \mathbb{T})$ be the set of timed languages, accepted by timed safety automata with k states and n clocks over the time domain $\mathbb{T}$. The following table summarizes the hierarchies for which one parameter is fixed (see Theorems 3.9, 3.10, 3.13, 4.4, 4.5, and Corollary 3.12).

$\mathcal{U}(k,n,\mathbb{N}) \subsetneq \mathcal{U}(k+1,n,\mathbb{N})$	$\mathcal{U}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \mathcal{U}(k+1,n,\mathbb{R}_{\geq 0})$
$\mathcal{U}(k,1,\mathbb{N}) \subsetneq \mathcal{U}(k,2,\mathbb{N}) = \mathcal{U}(k,n+2,\mathbb{N})$	$\mathcal{U}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \mathcal{U}(k,n+1,\mathbb{R}_{\geq 0})$
$\mathcal{T}(k,n,\mathbb{N}) \subsetneq \mathcal{T}(k+1,n,\mathbb{N})$	$\mathcal{T}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \mathcal{T}(k+1,n,\mathbb{R}_{\geq 0})$
$\mathcal{T}(k,n,\mathbb{N}) \subsetneq \mathcal{T}(k,n+1,\mathbb{N})$	$\mathcal{T}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \mathcal{T}(k,n+1,\mathbb{R}_{\geq 0})$

The final table refers to the clock hierarchies for a variable number of states and the state hierarchies for a variable number of clocks (see Theorems 3.11, 4.1, 4.2, 4.3 and Corollaries 3.5 and 3.8).

$\bigcup_k \mathcal{U}(k,1,\mathbb{N}) = \omega$-regular	$\bigcup_k \mathcal{U}(k,1,\mathbb{R}_{\geq 0}) = \omega$-regular
$\bigcup_n \mathcal{U}(k,n,\mathbb{N}) = k$-iocc	$\bigcup_n \mathcal{U}(1,n,\mathbb{R}_{\geq 0}) = \omega$-regular
$\bigcup_k \mathcal{T}(k,1,\mathbb{N}) =$ timed ω-reg.	$\bigcup_k \mathcal{T}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \bigcup_k \mathcal{T}(k,n+1,\mathbb{R}_{\geq 0})$
$\bigcup_n \mathcal{T}(k,n,\mathbb{N}) \subsetneq \bigcup_n \mathcal{T}(k+1,n,\mathbb{N})$	$\bigcup_n \mathcal{T}(k,n,\mathbb{R}_{\geq 0}) \subsetneq \bigcup_n \mathcal{T}(k+1,n,\mathbb{R}_{\geq 0})$

Notice that the restriction of one clock and constants at most 1 does not disable acceptance of any regular language with either time domain. Similarly, the restriction of one state and constants at most 1 does not disable acceptance of any regular language with the dense time domain. However, if we make the restriction of one clock and one state, and allow constants of arbitrary size, there are indeed regular languages that are not accepted. This is implied by the strictness of the hierarchies with one fixed parameter.

References

1. R. Alur, C. Courcoubetis, and T.A. Henzinger. The observational power of clocks. In *CONCUR 94*, LNCS 836, pp. 162–177. Springer-Verlag, 1994.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur and T.A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proc. 33rd FOCS*, pp. 177–186. IEEE Computer Society Press, 1992.
4. R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory in Practice*, LNCS 600, pp. 74–106. Springer-Verlag, 1992.
5. R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
6. R. Alur and T.A. Henzinger. Real-time system = discrete system + clock variables. In *Proc. of the First AMAST Workshop on Real-time Systems*, 1993. To appear.
7. C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Proceedings of FORTE '94*, 1994.
8. T.A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.
9. T.A. Henzinger and P. Kopke. Verification methods for the divergent runs of clock systems. In *FTRTFT 94*, LNCS 863, pp. 351–372. Springer-Verlag, 1994.
10. T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *ICALP 92*, LNCS 623, pp. 545–558. Springer-Verlag, 1992.
11. 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.
12. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pp. 133–191. Elsevier Science Publishers (North-Holland), 1990.