The Expressive Power of Clocks*

Thomas A. Henzinger† Peter W. Kopke Howard Wong-Toi

Computer Science Department
Cornell University, Ithaca, NY 14853

(tah|pkpk|howard)@cs.cornell.edu

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* [HNSY94]. Timed safety automata are timed automata [AD94] 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 [AH93b, DOY94, HNSY94, HK94]. It has been argued that with the explicit consideration of time, acceptance conditions are no longer useful abstractions to enforce liveness [Hen92], 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.

†Phone: (607) 255-3009. FAX: (607) 255-4428

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 [HNSY94], and therefore the timed expressiveness of timed safety automata is strictly less than that of timed automata [AD94]. 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* [AH92b, AH93a]. 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 [AH92a]. An infinite hierarchy of timed expressiveness is obtained, based upon the number of alternations. Clock hierarchies of a different nature are studied in [ACH94]. 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 [Tho90].

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 $(s, a) \stackrel{b}{\Rightarrow} (t, b)$ 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 a nonnegative integer. A *closed clock constraint* is a conjunction of formulas of the form $x \leq n$ and $x \geq n$. A *clock valuation* $\nu : C \to \mathbb{T}$ 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 $\mathcal{V}_{\mathbb{T}}^C$. It is naturally isomorphic to $\mathbb{T}^{|C|}$. Closed clock constraints specify closed subsets of $\mathcal{V}_{\mathbb{T}}^C$. 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 $\mathcal{V}_{\mathbb{T}}^C$ 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 $\mathcal{V}_{\mathbb{T}}^C$ 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* [HNSY94].

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 $\mathbb{T}$-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 $(\mathcal{A}, \mathcal{T})$ for $(\overline{a}, \overline{\delta})$. The *timed language* $[\![\mathcal{A}, \mathcal{T}]\!]^t_{\mathbb{T}}$ *accepted by* $(\mathcal{A}, \mathcal{T})$ under time domain $\mathbb{T}$ is the set of $\mathbb{T}$-timed words accepted by $(\mathcal{A}, \mathcal{T})$. The *untimed language* $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{T}}$ *accepted by* $(\mathcal{A}, \mathcal{T})$ 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 $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}} = [\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{R}_{\geq 0}}$ [HMP92]. 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 [AH92b, AH93a]. 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 [AD94].

Let h be the largest constant appearing in the clock constraints of $\mathcal{T}$. Define an equivalence relation $\equiv_h$ on the set $\mathcal{V}^C_{\mathbb{T}}$ 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* $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{T}}$ for $(\mathcal{A}, \mathcal{T})$ is the finite automaton $(S \times (\mathcal{V}^C_{\mathbb{T}}/\equiv_h), A, \rightarrow', S_0 \times \{\overline{\lambda x.0}\})$. 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 $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{T}}$ satisfying $\mathcal{B}$, and vice versa. The region automaton is the main tool for the analysis of timed safety automata [AD94].

Example.

Figure 1 depicts a timed safety automaton $(\mathcal{A}, \mathcal{T})$. The untimed language $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{T}}$ is the set of all sequences $\overline{\sigma} \in (a + b + c)^\omega$ with infinitely many b events and infinitely many c events.

3 Untimed Languages

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

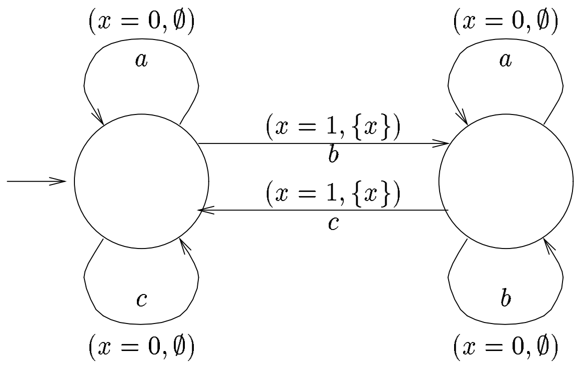

Figure 1: A timed safety automaton

3.1 Time Enough for Liveness

We give several ways to replace acceptance conditions on finite automata, or fairness conditions on event-recording 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 nonnegative 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 be 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." We then use two clocks 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. An acceptance condition may allow finitely many "bad transitions." We allow bad transitions to occur iff no time has passed since the beginning of a run. We do the case of Streett acceptance in detail, and sketch the argument for the remaining cases.

Let $\mathcal{X}$ be a Streett acceptance condition. Observe that a run $r = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ of $\mathcal{A}$ satisfies $\mathcal{X}$ 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 $J_{i,j+1 \bmod m(i)}$. We must store the two values i and j, corresponding respectively to the choice of $\mathcal{X}_{fin}$ and the next J to be visited. Of course, storing two values is no harder than storing one.

Let $\gamma : \{(i,j) | 1 \leq i \leq n \wedge 1 \leq j \leq m(i)\} \rightarrow \{1, \ldots, n|X|\}$ 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 $t \in J_{i,j}$,

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

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

where we take $\{1, \ldots, m(i)\}$ to be the range of mod $m(i)$. This completes the construction.

Note that if we wished to disallow time steps of duration zero from our dense-time model, the proof would need only minor modifications.

For the other acceptance conditions, we only indicate the decomposition into helpful transitions and the range of values to be stored by the two clocks. The latter gives the largest constant needed in clock constraints. For a generalized Büchi acceptance condition $\mathcal{X} = \{B_1, \ldots, B_n\}$, the helpful transitions are entries to the B_i. If entry to B_i was the last helpful transition, then entry to $B_{i+1 \bmod n}$ is the next. A range of $|\mathcal{X}|$ values needs to be stored. Muller acceptance is similar to Streett acceptance. First the infinite visitation set $M \in \mathcal{X}$ must be guessed, and then infinitely many visits to each $m \in M$ must be forced. So both the particular $M \in \mathcal{X}$ ($|\mathcal{X}|$ possible values), and the next $m \in M$ ($|S|$ possible values) must be stored. This gives a range of $|S||\mathcal{X}|$ values. Finally, Rabin acceptance requires simply guessing an $(I, J) \in \mathcal{X}$ ($|\mathcal{X}|$ possible values) such that I is visited infinitely often and J is not. The helpful transitions are visits to I. ∎

Fairness conditions do not require infinitely many visits to a particular element of a "good" set, merely infinitely many visits to the set itself. This is exactly what the previous construction forces of all divergent runs. With this modified definition of acceptance (infinitely many visits to the "good" sets, finitely many visits to the "bad" sets), a weak-fairness condition is a generalized Büchi acceptance condition, and a strong-fairness condition is a Streett acceptance condition.

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 $[\![\mathcal{A},\mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A},\mathcal{X}]\!]$.*
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 $[\![\mathcal{A},\mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A},\mathcal{S}]\!]$.*

Proof. 1. A value in $\{1, ..., N\}$ may be stored in $1 + \lceil \log_2 N \rceil$ clocks by using $\lceil \log_2 N \rceil$ clocks to store its binary representation, and one additional clock y to mediate increments. If N is not a power of two, then the encoding is especially simple. The bit vector $b_1 b_2 \dots b_m$ is encoded by the clock constraint $\bigwedge_{b_i=1} x_i \geq 1 \wedge \bigwedge_{b_i=0} x_i = 0$. To change the value stored from bit vector $b_1 b_2 \dots b_m$ to bit vector $c_1 c_2 \dots c_m$, use the atomic clock command

$$\left(y = 1 \ \wedge \bigwedge_{b_i=1} x_k \geq 2 \ \wedge \bigwedge_{b_i=0} x_k = 1, \{y\} \cup \{x_k \mid c_k = 0\} \right).$$

The bounds on the number of clocks result from the ranges of values required in the proof of Theorem 3.1.

We give the proof for Muller acceptance. Let $\mathcal{A} = (S, A, \rightarrow, S_0)$. Let the elements of $\mathcal{X}$ be enumerated $M_1, M_2, \dots, M_{|\mathcal{X}|}$. Let the elements of each M_i be enumerated $s_{i,1}, s_{i,2}, \dots, s_{i,m(i)}$. In this case, two numbers need be stored: the infinite visitation set $M_i \in \mathcal{X}$, and the next element $s_{i,j}$ of M_i to visit. There are therefore no more than $|\mathcal{X}||S|$ values in question. Let $N = |\mathcal{X}||S|$, and let $n = \lceil \log_2 N \rceil$. Assume for simplicity that N is not a power of two. A slightly modified construction is otherwise required. We will use n clocks $x_1, \dots, x_n$ in addition to y. For $k \leq N$ and $l \leq n$, let $bit(k,l)$ be the lth bit of the n-bit binary representation of k. Let $\gamma \colon \{(i,j) | 1 \leq i \leq |\mathcal{X}| \wedge 1 \leq j \leq m(i)\} \rightarrow \{k \in \mathbb{N} \mid 1 \leq k \leq |S||\mathcal{X}|\}$ be any one-to-one map. For each i, let $Bad_i = S \setminus M_i$. To each transition $s \xrightarrow{a} t$ we assign four classes of atomic clock commands. For the finite prefix before choice of the infinite visitation set, we have $(y = 0 \wedge \bigwedge_{i=1}^{n} x_i = 0, \emptyset)$. To guess M_i, we have for each i with $1 \leq i \leq |\mathcal{X}|$, the atomic clock command

$$\left(y = 1 \wedge \bigwedge_{j=1}^{n} x_j = 1, \{y\} \cup \{x_k \mid bit(\gamma(i,1), k) = 0\} \right).$$

Once M_i is chosen, we must disallow further visits to Bad_i. We must also not let time advance until the next $s_{i,j}$ is reached. For each i, j with $t \notin Bad_i$ and $t \neq s_{i,j}$, we have the atomic clock command

$$\left(y = 0 \ \wedge \bigwedge_{bit(\gamma(i,j),k)=1} x_k \geq 1 \ \wedge \bigwedge_{bit(\gamma(i,j),k)=0} x_k = 0, \emptyset \right).$$

Finally, if $t = s_{i,j}$, we have the atomic clock command

$$\left(y = 1 \ \wedge \bigwedge_{bit(\gamma(i,j),k)=1} x_k \geq 2 \ \wedge \bigwedge_{bit(\gamma(i,j),k)=0} x_k = 1, \{y\} \cup \{x_k \mid bit(\gamma(i, j+1 \bmod m(i)), k) = 0\} \right).$$

2. Let $(I_1, J_1), \ldots, (I_n, J_n)$ enumerate the elements of $\mathcal{X}$. We use one clock x_i for each element (I_i, J_i) of $\mathcal{X}$. For each transition $s \xrightarrow{a} t$, the edge constraint $\psi(s,a,t)$ consists of a single atomic clock command $(\varphi(s,a,t), X(s,a,t))$. The guard $\varphi(s,a,t)$ is $\bigwedge_{t \in I_i \setminus J_i}(x_i \leq 1)$. The reset set $X(s,a,t)$ is $\{x_i : t \in J_i\}$. The requirement $x_i \leq 1$ on every entry to I_i acts as a gate. If time is to diverge, infinitely many visits to I_i require infinitely many resets of x_i, and hence infinitely many visits to J_i.

Suppose $(r, \overline{\nu})$ is a divergent execution for the $\mathbb{T}$-timed word $(\overline{a}, \overline{\delta})$. If infinitely many of the states of r lie in $I_j \setminus J_j$, then the transition guards and the divergence of time imply that x_j is reset infinitely many times. Since x_j is reset only upon entry to J_j, infinitely many of the s_i lie in J_j. Therefore r satisfies the Streett condition $\mathcal{S}$.

Conversely, suppose $r = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ is a run of $\mathcal{A}$ satisfying $\mathcal{S}$. Let $\mathcal{S}_{fin}$ be the subset of $\mathcal{S}$ consisting of all pairs (I, J) such that only finitely many of the s_i lie in I. Let $\mathcal{S}_{inf} = \mathcal{S} \setminus \mathcal{S}_{fin}$. Let n_0 be large enough so that for all $n \geq n_0$, and all $(I, J) \in \mathcal{S}_{fin}$, $s_n \notin I$. If $\mathcal{S}_{inf}$ is empty, then let one unit of time pass with each transition after the n_0th, i.e., define $\delta_i = 1$ if $i > n_0$, and $\delta_i = 0$ otherwise. Since each transition guard $\varphi(s_k, a_k, s_{k+1})$ with $k > n_0$ is *true*, the valuation sequence $\overline{\nu}$ induced by r and $\overline{\delta}$ is such that $(r, \overline{\nu})$ is a divergent execution for the $\mathbb{T}$-timed word $(\overline{a}, \overline{\delta})$. If $\mathcal{S}_{inf}$ is nonempty, let $\overline{n} = n_1, n_2, \ldots$ be an increasing sequence of integers greater than n_0 such that for each $i > 0$, and for each $(I, J) \in \mathcal{S}_{inf}$, there is a j with $n_i < j < n_{i+1}$ such that $s_j \in J$. Such a sequence exists because r satisfies the Streett condition $\mathcal{S}$. Define a timing $\overline{\delta}$ by $\delta_i = 1$ if $i = n_j$ for some $j > 0$, and $\delta_i = 0$ otherwise. Let $\overline{\nu}$ be the valuation sequence induced by r and $\overline{\delta}$. We now show that each transition guard $\varphi(s_k, a_k, s_{k+1})$ is satisfied by $\nu_k + \delta_k$. If clock x_i corresponds to (I_i, J_i) in $\mathcal{S}_{fin}$, then the value of x_i is only constrained by transition guards $\varphi(s_k, a_k, s_{k+1})$ with $k \leq n_0$. Since $\delta_j = 0$ for all $j \leq n_0$, clock x_i has value 0 whenever it is constrained, and does not falsify any transition guard. By definition of $\overline{n}$, J_i is met (and hence x_i is reset) in between each pair of transitions advancing time. So if clock x_i corresponds to (I_i, J_i) in $\mathcal{S}_{inf}$, then x_i never attains a value greater than one. Therefore the value of x_i does not falsify any transition guard. ■

Since every Büchi acceptance condition is also a Streett acceptance condition, we have the following corollary.

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 $[\![\mathcal{A}, \mathcal{F}]\!] = [\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{T}}$.*

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 $[\![\mathcal{C}, \mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A}, \mathcal{X}]\!]$.*

Proof. For every ω-regular language $\mathcal{L}$, there is a finite automaton $\mathcal{C}$ and a Streett acceptance condition $\mathcal{X}$ with one element such that $[\![\mathcal{C}, \mathcal{X}]\!] = \mathcal{L}$. Hence the result follows from Theorem 3.3. ■

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 $[\![\mathcal{C}, \mathcal{T}]\!]_{\mathbb{T}} = [\![\mathcal{A}, \mathcal{F}]\!]$.*

Proof. For weak-fairness, the proof is the same as the proof that every generalized Büchi acceptance condition can be simulated by a standard Büchi acceptance condition by using several copies of the state space. Let $\mathcal{F} = \{a_1, \ldots, a_n\}$. We use n copies of $\mathcal{A}$. While control resides in copy i, no time is allowed to pass. Control passes to copy $i + 1 \bmod n$ upon an a_i transition, at which point one unit of time must pass. Let $\mathcal{A} = (S, A, \rightarrow, S_0)$. The automaton $\mathcal{C}$ has components $(S \times \{1, \ldots n\}, A, \rightarrow', S \times \{1\})$. For each transition $s \xrightarrow{a} t$ in $\mathcal{A}$, and for each $i \in \{1, \ldots, n\}$, we have one transition in $\mathcal{C}$. If $a \neq a_i$ and $s \in \mathsf{enabled}(a_i)$, then $(s, i) \xrightarrow{a}{}' (t, i)$ in $\mathcal{C}$ with the atomic clock command $(x = 0, \emptyset)$. If $a = a_i$ or $s \notin \mathsf{enabled}(a_i)$, then $(s, i) \xrightarrow{a}{}' (t, i+1 \bmod n)$ in $\mathcal{C}$ with the atomic clock command $(x = 1, \{x\})$. Time diverges iff there are infinitely many transitions from level i to level $i + 1 \bmod n$ for each $i \in \{1, \ldots, n\}$ iff for each $a_i \in \mathcal{F}$, either a_i is taken infinitely often or a_i is disabled infinitely often.

For strong-fairness, zero time is used to guess the complement of the set $\mathcal{X}_i$ from the proof of Theorem 3.1, and the finite prefix containing all entries to Bad_i. From this point, the acceptance condition is reduced to a generalized Büchi condition. Let $\mathcal{Y}_1, \ldots, \mathcal{Y}_n$ enumerate the subsets of $\mathcal{F}$. For $i \in \{1, \ldots, n\}$, define $Good_i = \{s \in S \mid \forall a \in \mathcal{F} \setminus \mathcal{Y}_i . s \notin \mathsf{enabled}(a)\}$, and let $\{a_{i,j} \mid 1 \leq j \leq |\mathcal{Y}_i|\}$ enumerate the elements of $\mathcal{Y}_i$. The automaton $\mathcal{C}$ has components $(S', A, \rightarrow', S_0)$. The state space S' for $\mathcal{C}$ is $S \cup \{(s, i, j) \in S \times \{1, \ldots, n\} \times \mathbb{N} \mid 1 \leq j \leq |\mathcal{Y}_i| \text{ and } s \in Good_i\}$.

For each transition $s \xrightarrow{a} t$ in $\mathcal{A}$, we have four classes of transitions in $\mathcal{C}$. For the finite prefix before $\mathcal{Y}_i$ is guessed, we have $s \xrightarrow{a}{}' t$ with atomic clock command $(x = 0, \emptyset)$. For each $i \in \{1, \ldots, n\}$, if $t \in Good_i$, then to guess $\mathcal{Y}_i$ we have $s \xrightarrow{a}{}' (t, i, 1)$ with atomic clock command $(x = 0, \emptyset)$. For each $i \in \{1, \ldots, n\}$, and each $j \in \{1, \ldots, |\mathcal{Y}_i|\}$, if $s, t \in Good_i$, we have two types: if $a \neq a_{i,j}$, then we have $(s, i, j) \xrightarrow{a}{}' (t, i, j)$ with atomic clock command $(x = 0, \emptyset)$; if $a = a_{i,j}$, then we have $(s, i, j) \xrightarrow{a}{}' (t, i, j + 1 \bmod |\mathcal{Y}_i|)$ with atomic clock command $(x = 1, \{x\})$. ■

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. Let $\mathcal{A} = (S, A, \rightarrow, S_0)$, let $\mathcal{C} = (S', A, \rightarrow', S'_0)$ be a finite automaton, and let $\mathcal{B}'$ be a standard Büchi acceptance condition for $\mathcal{C}$ such that $[\![\mathcal{C}, \mathcal{B}']\!] = L$. While performing a computation on $\mathcal{A}$, we simulate the computation of $\mathcal{C}$ by using one clock $x_{s'}$ for each state s' of $\mathcal{C}$. The clock with the largest value corresponds to the current state of $\mathcal{C}$. The current state of $\mathcal{C}$ 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 $\mathcal{C}$. No transition is enabled if $x_{\mathcal{B}'}$ is greater than one, and $x_{\mathcal{B}'}$ is reset only when control enters $\mathcal{B}'$.

Let $s' \xrightarrow{a}{}' t'$ be a transition in $\mathcal{C}$. To record this transition with clocks, we define the atomic clock command $(global \wedge In_{s'}, Reset_{t'} \cup P'_{t'})$. The clock constraint *global* is $x_{\mathcal{B}'} \leq 1$. The reset set $Reset_{t'}$ is $\{x_{u'} | u' \in S' \text{ and } u' \neq t'\}$; $P'_{t'} = \{x_{\mathcal{B}'}\}$ if $t' \in \mathcal{B}'$, otherwise $P'_{t'} = \emptyset$.

Let $s \xrightarrow{a} t$ be a transition in $\mathcal{A}$. Let the clock command $\zeta(s, a, t)$ be the set of all atomic clock commands $(global \wedge In_{s'}, Reset_{t'} \cup P'_{t'})$ such that $s' \xrightarrow{a}{}' t'$ is a transition of $\mathcal{C}$ (on the same event a). For each initial state $s \in S_0$, and each transition $s \xrightarrow{a} t$, we must allow the first step of a computation to begin with $s \xrightarrow{a} t$. Let *Init* be the clock constraint $\bigwedge_{s',t' \in S'} (s' = t' \wedge t' > 0)$. Let the clock command $\xi(s, a, t)$ be the set of all atomic clock commands $(Init \wedge global, Reset_{t'})$ such that $s' \xrightarrow{a}{}' t'$ is a transition of $\mathcal{C}$ and $s' \in S'_0$. Finally, for any transition $s \xrightarrow{a} t$ of $\mathcal{A}$, define

$$\psi(s, a, t) = \begin{cases} \zeta(s, a, t) & \text{if } s \notin S_0, \\ \zeta(s, a, t) \cup \xi(s, a, t) & \text{if } s \in S_0. \end{cases}$$

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 untimed 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.*

Proof. We need some additional notation for this proof. A region equivalence class $\overline{\mu}$ of clock valuations assigns to each clock x a value in $\{0, (0,1), 1, (1,2), \ldots, (h-1, h), h, \infty\}$, where h is the largest constant used in clock constraints. This is done by $\overline{\mu}(x) = m$ if for every clock valuation $\nu \in \overline{\mu}$, $\nu(x) = m$; $\overline{\mu}(x) = (m, m+1)$ if for every clock valuation $\nu \in \overline{\mu}$, $\nu(x) \in (m, m+1)$; and $\overline{\mu}(x) = \infty$ if for every clock valuation $\nu \in \overline{\mu}$, $\nu(x) > h$.

For each $k, n \in \mathbb{N}$, there are numbers $f(k,n)$ and $g(k,n)$ such that the untimed language $L = a^{g(k,n)}(b^{1+f(k,n)})^* c^\omega$ is accepted by no timed safety automaton with k states and n clocks, but is accepted by a timed safety automaton with one additional state. The number $f(k,n)$ is the length of the largest simple cycle $(s_1, \overline{\mu}_1), (s_2, \overline{\mu}_2), \ldots (s_N, \overline{\mu}_N)$ in a region graph of a k-state n-clock automaton with the following two properties: (1) repeatedly looping through the cycle forces time to converge, (i.e., the infinite path around the cycle does not satisfy the Büchi acceptance condition of the region automaton) and (2) the cycle admits no "cross edges", i.e., for every p, q, if there is an edge from $(s_p, \overline{\mu}_p)$ to $(s_q, \overline{\mu}_q)$ in the region automaton, then $q = p + 1 \bmod N$. A cycle with these two properties is called *purely converging.* The number $f(k,n)$ does not depend on the size of constants in clock constraints, but only on the number of states and clocks. To see this, notice that any cycle with this property has the following property: for every clock x, either $\overline{\mu}_p(x) = \overline{\mu}_q(x)$ for all $1 \le p, q \le N$, or for every $1 \le p \le N$, $\overline{\mu}_p(x) = 0$ or $\overline{\mu}_p(x) = (0,1)$. Any cycle without this property takes more than one time unit to traverse. It follows that there is a maximal simple purely converging cycle $(s_1, \overline{\mu}_1), (s_2, \overline{\mu}_2), \ldots, (s_{f(k,n)}, \overline{\mu}_{f(k,n)})$ such that for every clock x, and for every $1 \le p \le f(k,n)$, $\overline{\mu}_p(x) = 0$ or $\overline{\mu}_p(x) = (0,1)$.

We first show that no timed safety automaton with k states and n clocks accepts L. Suppose $(\mathcal{A}, \mathcal{T})$ is such an automaton accepting L. Let $(r, \overline{\nu})$, where $r = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$, be a divergent execution of $(\mathcal{A}, \mathcal{T})$ generating the word $a^{g(k,n)} b^{m(1+f(k,n))} c^\omega$, where m is larger than the number of states in the region automaton $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{R}_{\ge 0}}$. Then $(s_0, \overline{\nu}_0) \xrightarrow{a_0} (s_1, \overline{\nu}_1) \xrightarrow{a_1} \cdots$ is a run of $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{R}_{\ge 0}}$, where $\overline{\nu_i}$ is the region of ν_i. There are then indices $i < j$ with $g(k,n) \le i < j \le g(k,n) + m(1 + f(k,n))$ such that $s_i = s_j$. Then $j - i$ is a multiple of $1 + f(k,n)$, for otherwise by pumping the substring, $(\mathcal{A}, \mathcal{T})$ accepts a word not in L. Also, the cycle from s_i to s_j is time converging, i.e., it does not satisfy the generalized Büchi acceptance condition on $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{R}_{\ge 0}}$, for otherwise $(\mathcal{A}, \mathcal{T})$ accepts $a^{g(k,n)} b^\omega$. Therefore, by definition of $f(k,n)$, this cycle must have a cross edge. Hence there exists a strictly shorter subcycle contained in $s_i, \ldots, s_j$. This subcycle is time convergent for the same reason. And again, its length must be a multiple of $1 + f(k,n)$. Therefore this subcycle must have a cross edge. In this manner, we obtain a strictly decreasing infinite sequence of natural numbers (the lengths of the subcycles). No such sequence exists.

Let $(\mathcal{A}, \mathcal{T})$ be a k-state n-clock timed safety automaton with a maximal purely converging cycle $(s_1, \overline{\mu}_1), (s_2, \overline{\mu}_2), \ldots, (s_{f(k,n)}, \overline{\mu}_{f(k,n)})$, in its region automaton, such that for every clock x_i, and every $1 \le p \le f(k,n)$, $\overline{\mu}_p(x_i) = 0$ or $\overline{\mu}_p(x_i) = (0,1)$. We build a $(k+1)$-state n-clock automaton accepting the language $a^{g(k,n)}(b^{1+f(k,n)})^* c^\omega$, where the number $g(k,n)$ will be determined by the construction. Consider Figure 2. The left hand side depicts the maximal purely converging cycle in the region automaton $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{R}_{\ge 0}}$. The right hand side is the region automaton of a timed safety automaton $(\mathcal{C}, \mathcal{T}')$ built from $(\mathcal{A}, \mathcal{T})$ by the addition of one state. The extra state is used for two purposes: (1) to extend the purely converging cycle by one, and (2) for bootstrapping the automaton for entry to the cycle. The clock commands ψ and $\{(\varphi_2, X_2)\}$ accomplish the bootstrapping. One atomic clock command $(\bigwedge_i x_i = 2, \emptyset) \in \psi$ allows arbitrarily many a events at the start of a run. The other atomic clock commands composing ψ are used in turn to reset the clocks so that their fractional parts are correctly ordered upon entry into $(s, \overline{\mu})$. For example, if $0 = \overline{\mu}(x) < \overline{\mu}(y) < \overline{\mu}(z)$, then $\psi = \{(x = y = z = 2, \emptyset), (2 < x = y = z < 3, \{z\}), (2 < x = y < 3 \wedge 0 < z < 1, \{y\})\}$ and $(\varphi_2, X_2) = (2 < x < 3 \wedge 0 < y < z < 1, \{x\})$. The number $g(k,n)$ is then the number of these bootstrapping transitions. Except in trivial cases, any maximal purely converging cycle has some node with some clock having value zero. Such a node may serve as $(s, \overline{\mu})$. The c-transition with atomic clock command $(\bigwedge_i x_i \ge 3, \emptyset)$ then gives the infinite suffix of c events. ■

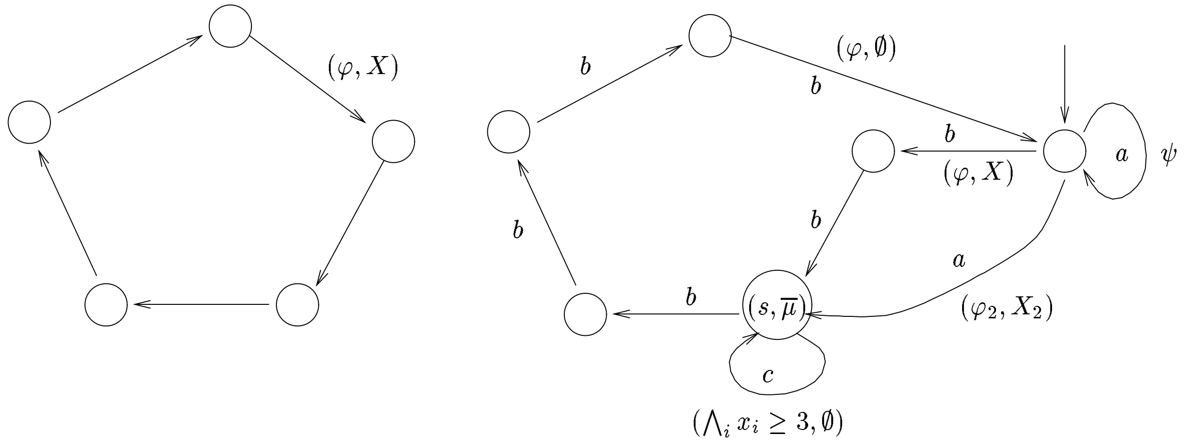

Figure 2: Extending a purely converging cycle with an additional state

The clock hierarchy for a fixed number of states.

Theorem 3.10 *For every $k, n \in \mathbb{N}$, the class of untimed 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.*

Proof. This proof follows the same path as the proof of Theorem 3.9. For each $k, n \in \mathbb{N}$, there is a number $h(k,n)$ such that the language $L = a^{h(k,n)}(b^{1+f(k,n)})^* c^\omega$ is accepted by no timed safety automaton with k states and n clocks, but is accepted by a timed safety automaton with one additional clock. The number $f(k,n)$ is defined is in the proof of Theorem 3.9. The proof that no k-state, n-clock automaton accepts L is as before.

Let $(\mathcal{A}, \mathcal{T})$ be a k-state n-clock timed safety automaton with a maximal purely converging cycle $(s_1, \overline{\mu}_1), (s_2, \overline{\mu}_2), \ldots, (s_{f(k,n)}, \overline{\mu}_{f(k,n)})$ in its region automaton, such that for every clock x_i, and every $1 \leq p \leq f(k,n)$, $\overline{\mu}_p(x_i) = 0$ or $\overline{\mu}_p(x_i) = (0,1)$. We build a k-state $(n+1)$-clock automaton $(\mathcal{C}, \mathcal{T}')$ accepting $a^{h(k,n)}(b^{1+f(k,n)})^* c^\omega$. See Figure 3. Notice that except in trivial cases, no maximal purely converging cycle can contain any node $(t, \overline{\nu})$ with clock region $\overline{\nu}$ mapping each clock x to ∞. The automaton $(\mathcal{C}, \mathcal{T}')$ uses one clock y in addition to the clocks of $\mathcal{A}$. The set of initial states of $\mathcal{C}$ is $\{s\}$. The clock x_i is any clock taking a nonzero value at some node on the cycle. The clock command ψ_a is similar to ψ from the previous construction. It uses the atomic clock command $(y = 2 \wedge \bigwedge_i x_i = 2, \emptyset)$ to allow a prefix consisting of arbitrarily many a events. In addition, it has bootstrapping commands as above, with y the first clock reset to 0. The clock command $\psi_c = \{(y > 3, \emptyset)\}$ produces the infinite suffix of c events. Every edge around the cycle inherited from $(\mathcal{A}, \mathcal{T})$ has the additional constraint $y \leq 1$. The first bootstrapping command cannot be repeated between loops around the cycle because no node on the cycle may have the clock region $\lambda x.0$ or the clock region defined by $\bigwedge_{i,j} 0 < x_i = x_j < 1$ (otherwise the cycle satisfies the Büchi acceptance condition on the region automaton of $(\mathcal{A}, \mathcal{T})$). The following bootstrap commands cannot be repeated because each requires a gap of greater than one between some pair of clocks.

The case $n = 1$ requires special handling, as does $k = n = 1$. The details are left to the reader. ■

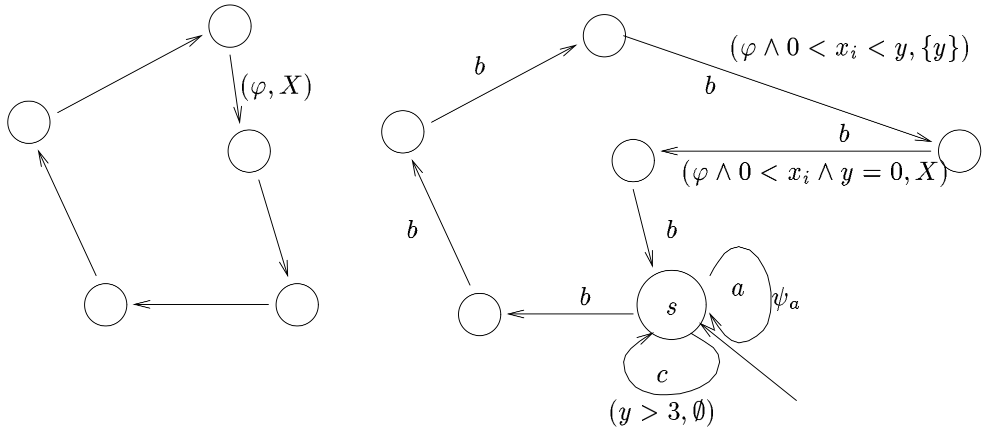

Figure 3: Extending a purely converging cycle with an additional clock

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 (iocc)* 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. Notice that the only equivalence relation with multiplicity 1 is equality, which induces an *infinitely often change state* acceptance condition.

Theorem 3.11 *For every $k \geq 1$, the set of untimed 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. First we show that the untimed language of every timed safety automaton with k states over discrete time is a k-iocc language. Over the discrete time domain, the divergence of time is equivalent to infinitely many steps taking nonzero time. We would like divergence to be equivalent to infinitely many changes of region. Infinitely many region changes does guarantee time divergence, because only a fixed finite number of region changes corresponding to steps taking zero time may occur between region changes corresponding to steps taking nonzero time. This is because each such zero-time region change corresponds to resetting one or more clocks to zero. But time divergence does not imply infinitely many region changes. For any region in which each clock is either 0 or larger than all relevant constants can perform a nonzero time step and return to itself. Therefore, we duplicate every such state $(s, \overline{v})$ in the region graph corresponding to such regions. Transitions from $(s, \overline{v})$ to itself taking nonzero time shuttle between the two copies.

Let $(\mathcal{A}, \mathcal{T})$ be a timed safety automaton. Let $Reg'(\mathcal{A}, \mathcal{T})_\mathbb{N}$ be the region automaton for $(\mathcal{A}, \mathcal{T})$ modified as described above. The state space of $Reg'(\mathcal{A}, \mathcal{T})_\mathbb{N}$ is contained in $S \times (\mathcal{V}_\mathbb{N}^C / \equiv_h) \times \{0, 1\}$, where S is the state space of $\mathcal{A}$, and $\mathcal{V}_\mathbb{N}^C / \equiv_h$ is the set of clock valuations modulo region equivalence. Define $(s,\overline{\nu},i) \sim (t,\overline{\mu},j)$ iff $\overline{\nu} = \overline{\mu}$ and $i = j$. Then $\sim$ has multiplicity k, and $[\![\mathcal{A},\mathcal{T}]\!]_{\mathbb{N}} = [\![Reg'(\mathcal{A},\mathcal{T})_{\mathbb{N}},\mathcal{B}_{\sim}]\!]$.

Now we show that every k-iocc language is the language of a k-state timed safety automaton over discrete time. Let $\mathcal{A} = (S, A, \rightarrow, S_0)$ be a finite automaton, and let $\sim$ be an equivalence relation of multiplicity k on S. Let the equivalence classes be denoted $S_1,\ldots,S_n$, and let the elements of class S_i be $s_{i,1},\ldots,s_{i,m(i)}$. We build a timed automaton $(\mathcal{C},\mathcal{T})$ with k states $t_1,\ldots,t_k$ such that $[\![\mathcal{C},\mathcal{T}]\!]_{\mathbb{N}} = [\![A,\mathcal{B}_{\sim}]\!]$. We use two clocks x and y to encode the current class S_i, and the state t_j to encode control residing in $s_{i,j}$. By allowing time to advance precisely when the equivalence class of $\mathcal{A}$ changes, we implement the iocc acceptance condition by the divergence of time.

For each transition $s_{i,j} \xrightarrow{a} s_{i',j'}$ in $\mathcal{A}$ where $i \neq i'$, we have a transition $t_j \xrightarrow{a}{}' t_{j'}$ in $\mathcal{C}$ labeled with the atomic clock commands

$$(x = i + i' \wedge y = i', \{x\}) \text{ and } (y = i + i' \wedge x = i', \{y\}).$$

For each transition $s_{i,j} \xrightarrow{a} s_{i,j'}$ in $\mathcal{A}$, we force time to stand still along a transition $t_j \xrightarrow{a}{}' t_{j'}$ in $\mathcal{C}$ with the atomic clock commands $(x = i \wedge y = 0, \emptyset)$ and $(y = i \wedge x = 0, \emptyset)$.

The set of initial states of $\mathcal{C}$ is $\{t_1\}$. For each initial state $s_{i,j} \in S_0$, and each transition $s_{i,j} \xrightarrow{a} s_{i',j'}$, we have a transition $t_1 \xrightarrow{a}{}' t_{j'}$ in $\mathcal{C}$ labeled by the atomic clock command $(x = i' \wedge y = i', \{y\})$. Such a transition may be taken only once, and so cannot affect the divergence of time. ■

The state hierarchy for a variable number of clocks.

Corollary 3.12 *For every $k \in \mathbb{N}$, there is an untimed 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 $\mathbb{N}$.*

Proof. Let $k \in \mathbb{N}$, $k \geq 1$, and let $a_1,\ldots,a_{k+1}$ be distinct events. We show that the ω-regular language $L = (a_1 \cdots a_{k+1})^* a_1^\omega$ is not a k-iocc language. Then by Theorem 3.11, L is not the language of any timed safety automaton with k states over discrete time. On the other hand, L is easily expressible as the language of a timed safety automaton with $k+1$ states over discrete time.

Suppose $\mathcal{A}$ is a finite automaton, and $\sim$ is an equivalence relation of multiplicity k such that $L = [\![\mathcal{A},\mathcal{B}_{\sim}]\!]$. Let

$$r = s_{1,1} \xrightarrow{a_1} s_{1,2} \xrightarrow{a_2} \cdots s_{1,k+1} \xrightarrow{a_{k+1}} s_{2,1} \xrightarrow{a_1} s_{2,2} \xrightarrow{a_2} \cdots s_{2,k+1} \xrightarrow{a_{k+1}} \cdots$$

be a run of $\mathcal{A}$ satisfying $\mathcal{B}_{\sim}$ and generating the word $(a_1 \cdots a_{k+1})^h a_1^\omega$, where h is greater than the number of states of $\mathcal{A}$. Let $(i,i'),(j,j')$ be the first pair of distinct indices with $s_{i,i'} = s_{j,j'}$. Then $i' = j'$, for otherwise omitting the portion of the run in between $s_{i,i'}$ and $s_{j,j'}$ leaves a run of $\mathcal{A}$ satisfying $\mathcal{B}_{\sim}$ that generates a word not in L. Since there are no duplicates among the more than k states $s_{i,i'}, s_{i,i'+1},\ldots,s_{j,i'-1}$, and since $\sim$ has multiplicity k, two of these states must belong to different equivalence classes. But this implies that

$$(s_{1,1} \xrightarrow{a_1} \cdots \xrightarrow{a_{i'-1}} s_{i,i'} \xrightarrow{a_{i'}})(s_{i,i'+1} \xrightarrow{a_{i'+1}} \cdots \xrightarrow{a_{i'-1}} s_{j,i'} \xrightarrow{a_{i'}})^\omega$$

is a run of $\mathcal{A}$ satisfying $\mathcal{B}_{\sim}$. This implies $(a_1 \cdots a_{k+1})^\omega \in L$, which is a contradiction. ■

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 fixed 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 $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}} = [\![\mathcal{A}, \mathcal{T}']\!]_{\mathbb{N}}$.*

Proof. The encoding is similar to that performed in the second half of Theorem 3.11. Recall that the state space of the discrete-time region automaton $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ is $S \times \mathcal{V}_{\mathbb{N}}^{C} / \equiv_h$, where S is the state space of $\mathcal{A}$, and $\equiv_h$ is an equivalence relation of finite index. Let the regions be enumerated $\overline{\nu}_1, ..., \overline{\nu}_n$. The region component of the state space of $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ will be encoded by two clocks x and y; region $\overline{\nu}_i$ is encoded by $|x - y| = i$.

For each transition $(s, \overline{\nu}_i) \xrightarrow{a} (t, \overline{\nu}_{i'})$ of $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ with $i \neq i'$, or with $i = i'$ if the transition corresponds to the passage of time in $(\mathcal{A}, \mathcal{T})$, let $\psi'(s, a, t)$ contain the atomic clock commands given in Theorem 3.11 for changing from value i to value i'.

For every transition $(s, \overline{\nu}_i) \xrightarrow{a} (t, \overline{\nu}_i)$ of $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ maintaining the same region, and corresponding to the passage of zero units of time, we force time to stand still with the atomic clock commands $(x = i \wedge y = 0, \emptyset)$ and $(y = i \wedge x = 0, \emptyset)$.

Some transitions $(s, \overline{\nu}_i) \xrightarrow{a} (t, \overline{\nu}_j)$ with $i \neq j$ in $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ may correspond to transitions taking no time in $(\mathcal{A}, \mathcal{T})$, while their versions in $(\mathcal{A}, \mathcal{T}')$ do take time. However this does not affect the divergence of time, because only finitely many such transitions may be taken in $Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}$ before time must pass again. This is because each such transition corresponds to the resetting of at least one clock to zero; so if m is the number of clocks used by $\mathcal{T}$, there can be no more than m such transitions without a time passage transition. We omit the bootstrapping commands required for the initial transition. ■

Corollary 3.14 *The set of k-iocc languages coincides with the set of languages accepted by timed safety automata with k states and 2 clocks over discrete time.*

For completeness, we show that two clocks are indeed better than one.

Proposition 3.15 *For every $k \geq 1$, the set of untimed languages accepted by timed safety automata with k states and 1 clock is properly contained in the set of untimed languages accepted by timed safety automata with k states and 2 clocks.*

Proof. We show that the untimed language $L = a^{k+1}b^{\omega}$ is not accepted by any k-state automaton with one discrete-time clock. On the other hand, L is easily seen to be accepted by a finite automaton with the infinitely often change state acceptance condition; hence L is accepted by a 1-state, 2-clock timed safety automaton over discrete time.

Suppose $(\mathcal{A}, \mathcal{T})$ is a timed safety automaton with one clock and k states such that $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}} = L$. Let $(r, \overline{\nu})$ be a divergent execution of $(\mathcal{A}, \mathcal{T})$ for the $\mathbb{N}$-timed word $(\overline{a}, \overline{\delta})$,where $r = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots$. Let x be the single clock of $\mathcal{T}$. Since $\mathcal{A}$ has k states, there exist i and j with $i < j \leq k$ such that $s_i = s_j$. Consider the values of the clock x as the transitions $s_i \xrightarrow{a} s_{i+1}$ and $s_j \xrightarrow{a} s_{j+1}$ are made, that is, $\nu_i(x) + \delta_i$ and $\nu_j(x) + \delta_j$. If $\nu_i(x) + \delta_i < \nu_j(x) + \delta_j$, then by passing more time in state s_i

the first time around, the transition to state s_{j+1} can be made prematurely. That is, the timing $\overline{\epsilon}$ defined by

$$\epsilon_n = \begin{cases} \delta_n, & \text{if } n < i \\ \nu_j(x) + \delta_j - \nu_i(x), & \text{if } n = i \\ \delta_{n+j-i}, & \text{if } n > i \end{cases}$$

and the run $s_0 \xrightarrow{a} \cdots s_i \xrightarrow{a} s_{j+1} \xrightarrow{a_{j+1}} \cdots$ of $\mathcal{A}$ make $a^{k-j+i}b^\omega$ an element of $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}}$. Similarly, if $\nu_i(x) + \delta_i \geq \nu_j(x) + \delta_j$, then by possibly passing more time in state s_j the second time around, the word $a^{k+j-i}b^\omega$ is seen to be in $[\![\mathcal{A}, \mathcal{T}]\!]_{\mathbb{N}}$. In either case, we have a contradiction. ■

4 Timed Languages

We study the expressiveness of timed safety 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-time 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. Intuitively, a timed safety automaton with n clocks can only record the delays associated with at most n different events, and so cannot enforce the correct timing constraints over $n+1$ concurrent events. Let L_{n+1} be the timed language consisting of all timed sequences $(\overline{a}, \overline{\delta})$ such that for each integer $i \geq 0$ 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 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, or else they cannot enforce events occurring with separations of precisely 1 time unit. Therefore at each time $1 + 1/i$ for $1 \leq i \leq n+1$ there is a clock with integer value. Since no clock may take integer value at more than one time instance in the time interval between time 1 and time 2, there must be at least $n+1$ clocks. ■

The discrete-time 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. The key idea is that all timed words accepted by a timed safety automaton over $\mathbb{N}$ can be captured by its finite region automaton. Delays between events are all integer valued and can be expressed by a single clock measuring the time elapsed since the last event.

Given a timed safety automaton $(\mathcal{A}, \mathcal{T})$, we construct a timing condition $\mathcal{T}'$ over one clock x such that $(Reg(\mathcal{A}, \mathcal{T})_{\mathbb{N}}, \mathcal{T}')$ accepts $[\![\mathcal{A}, \mathcal{T}]\!]^t_{\mathbb{N}}$. Let h be the value of the largest constant used in any comparison in $\mathcal{T}$. For each transition $s \xrightarrow{a} t$ in $\mathcal{A}$ there is a transition $(s, \overline{\nu}_i) \xrightarrow{a} (t, \overline{\nu}_j)$ in $Reg(\mathcal{A}, \mathcal{T})$ with atomic clock command $(x = m, \{x\})$ for $0 \leq m \leq h$, provided $(\overline{\nu}_i + m, \overline{\nu}_j) \in \psi(s, a, t)$. There is also a transition $(s, \overline{\nu}_i) \xrightarrow{a} (t, \overline{\nu}_j)$ with the atomic clock command $(x > h, \{x\})$, provided $(\overline{\nu}_i + h + 1, \overline{\nu}_j) \in \psi(s, a, t)$. ■

The state hierarchy for a variable number of clocks.

We now consider the hierarchy of expressiveness obtained by varying the number of states.

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.*

Proof. The timed language $(a, 0)^k (b, 1)^\omega$ is not accepted by any timed safety automaton with k states, but is accepted by one 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. When there are no 0 time steps, any k-state automaton $(\mathcal{A}, \mathcal{T})$ can be simulated by a 1-state automaton $(\mathcal{C}, \mathcal{T}')$ having k additional clocks $\{x_1, \ldots, x_k\}$. State i of $\mathcal{A}$ corresponds to the truth of the clock constraint $\bigwedge_{j \neq i} x_i > x_j$. 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, while keeping the number of clocks fixed, 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.*

Proof. Consider the language $L = \{(a^\omega, \overline{\delta})\}$, where for all $i \geq 0$, $\delta_i = 1$ if $i \bmod ((k+1)(n+1)+1) = 1$, and 0 otherwise. We claim it is accepted by a $(k + 1)$-state automaton with n clocks, but not by any k-state automaton with n clocks. Let $(\mathcal{A}_{k+1}, \mathcal{T})$ have clocks $x_1, x_2, \ldots, x_n$ and states $s_1, s_2, \ldots, s_{k+1}$ of which s_1 is initial. In any divergent execution, one time unit passes from state s_1, and then without any further time passing, control repeatedly cycles $k + 1$ times through the states, resetting one clock at the end of each cycle except the last cycle where the clock is reset at the beginning of the cycle. The final cycle ends at state s_1 with all clocks reset to 0. For $1 \leq i \leq k$ there is a transition $s_i \xrightarrow{a} s_{i+1}$ with clock commands $(\bigwedge_{p<q} x_p = 0 \wedge \bigwedge_{p \geq q} x_p = 1, \emptyset)$ for each $1 \leq q \leq n - 1$. There is a transition $s_{k+1} \xrightarrow{a} s_1$ with clock commands $(\bigwedge_{p<q} x_p = 0 \wedge \bigwedge_{p \geq q} x_p = 1, \{x_q\})$ for each $1 \leq q \leq n - 1$. The final cycle is encoded by transitions $s_{k+1} \xrightarrow{a} s_{k+1}$ with clock command $(\bigwedge_{p<n} x_p = 0 \wedge x_n = 1, \{x_n\})$, and for $1 \leq i \leq k$, $s_{i+1} \xrightarrow{a} s_i$ with clock command $(\bigwedge_{p \leq n} x_p = 0, \emptyset)$.

For a k-state n-clock automaton $(\mathcal{A}, \mathcal{T})$ from any state-valuation pair a run of the automaton may pass through at most $k(n + 1)$ different state-valuation pairs without letting any time pass. This corresponds to resetting each of the n clocks and visiting each state for each combination of reset clocks, as demonstrated in the above construction. Thus the run of $(\mathcal{A}, \mathcal{T})$ accepting the timed word in L must repeat a state-valuation pair while accepting the first $(k+1)(n+1)$ symbols, and hence will accept timed words with arbitrarily many events at time 1. ■

The clock hierarchy for a fixed number of states.

Similarly, increasing the number of clocks, while keeping the number of states fixed, strictly increases expressiveness. The proof of the following theorem is similar to the proof of Theorem 4.4.

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, Corollaries 3.12 and 3.14, and Proposition 3.15).

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

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).

$\bigcup_k \mathcal{U}(k,1,\mathbb{N})$	$=$	ω-regular	$\bigcup_k \mathcal{U}(k,1,\mathbb{R}_{\geq 0})$	$=$	ω-regular
$\bigcup_n \mathcal{U}(k,n,\mathbb{N})$	$=$	k-iocc	$\bigcup_n \mathcal{U}(1,n,\mathbb{R}_{\geq 0})$	$=$	ω-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

[ACH94] R. Alur, C. Courcoubetis, and T.A. Henzinger. The observational power of clocks. In B. Jonsson and J. Parrow, editors, *CONCUR 94: Concurrency Theory*, Lecture Notes in Computer Science 836, pages 162–177. Springer-Verlag, 1994.

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

[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.

[AH93a] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[AH93b] R. Alur and T.A. Henzinger. Real-time system = discrete system + clock variables. In T. Rus, editor, *Proceedings of the First AMAST Workshop on Real-time Systems*, 1993. To appear.

[DOY94] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Proceedings of FORTE '94*, 1994.

[Hen92] T.A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.

[HK94] T.A. Henzinger and P. Kopke. Verification methods for the divergent runs of clock systems. 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 351–372. Springer-Verlag, 1994.

[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(2):193–244, 1994.

[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 (North-Holland), 1990.